

USER GUIDE

# Essential Studio

## For TypeScript

---

Version - v18.1.0.36 | Release Date - March 19, 2020

|  |    |
|--|----|
| NuGet Packages .....                                 | 62 |
| Get the Syncfusion NuGet feed URL .....              | 62 |
| Add the Syncfusion NuGet feed URL.....               | 62 |
| Windows .....  | 62 |
| macOS .....  | 63 |
| Installing NuGet Packages .....                      | 64 |
| Using NuGet Package Manager.....                     | 64 |
| Using Package Manager Console .....                  | 65 |
| Accordion.....                                       | 65 |
| Configure Multiple Open .....                        | 68 |
| Customize Icon.....                                  | 73 |
| Header customization .....                           | 76 |
| Collapsible.....                                     | 76 |
| Enable Header expand.....                            | 77 |
| Set selected header .....                            | 79 |
| Multiple selection .....                             | 80 |
| AJAX settings.....                                   | 82 |
| Populate accordion with AJAX content .....           | 82 |
| View multiple contents .....                         | 84 |
| Enabling multiple panel open .....                   | 84 |
| Accordion Panel enabler .....                        | 85 |
| Enable/Disable widget .....                          | 85 |
| Enable panel items.....                              | 87 |
| State Persistence .....                              | 88 |
| Configure state persistence of Accordion panel ..... | 88 |
| Appearance and Styling .....                         | 89 |
| Adjusting Accordion size .....                       | 89 |
| Rounded corner .....                                 | 91 |
| Customize Accordion icon.....                        | 92 |
| Animations.....                                      | 94 |
| Expand and collapse speed .....                      | 94 |
| Theme.....   | 94 |
| CSS class.....                                       | 95 |
| RTL Support.....                                     | 97 |
| Enabling RTL Support .....                           | 97 |



|   |     |
|---|-----|
| Keyboard Navigation.....                          | 98  |
| Template Support .....                            | 100 |
| Autocomplete .....                                | 102 |
| Overview.....                                     | 102 |
| Key Features .....                                | 102 |
| Getting Started .....                             | 102 |
| Creating an AutoComplete in Typescript .....      | 102 |
| Data Binding.....                                 | 103 |
| Enable Popup Button .....                         | 104 |
| Data Binding.....                                 | 105 |
| Fields.....                                       | 105 |
| Multiple Columns.....                             | 107 |
| Example .....                                     | 108 |
| MultiSelection.....                               | 109 |
| Grouping.....                                     | 110 |
| Templates .....                                   | 111 |
| Validation.....                                   | 111 |
| Accessibility.....                                | 113 |
| Keyboard Interaction .....                        | 113 |
| Barcode.....                                      | 114 |
| Overview.....                                     | 114 |
| Getting Started .....                             | 114 |
| Create Barcode in Typescript .....                | 114 |
| BulletGraph.....                                  | 115 |
| Overview.....                                     | 115 |
| Getting Started .....                             | 116 |
| Adding Script Reference.....                      | 116 |
| Create your first BulletGraph in TypeScript ..... | 116 |
| Bullet Graph Dimensions .....                     | 124 |
| Size.....   | 124 |
| Value for performance bar.....                    | 124 |
| Comparative measure value .....                   | 125 |
| Theme.....  | 125 |
| Orientation.....                                  | 126 |
| Flow direction .....                              | 126 |

|  |     |
|--|-----|
| Qualitative range size.....                                  | 126 |
| Quantitative scale length .....                              | 127 |
| Bullet Graph Caption.....                                    | 127 |
| Title .....  | 127 |
| Subtitle.....  | 128 |
| Indicator.....   | 129 |
| Trim.....  | 130 |
| Text Placement .....   | 131 |
| Data Binding.....  | 134 |
| Local Data .....   | 134 |
| Remote Data .....  | 135 |
| Quantitative Scale.....                                      | 136 |
| Range for Quantitative Scale.....                            | 136 |
| Quantitative scale location .....                            | 136 |
| Major ticks .....  | 137 |
| Minor ticks .....  | 137 |
| Tick position .....  | 138 |
| Tick Placement .....   | 138 |
| Quantitative scale labels .....                              | 139 |
| Label Placement.....   | 140 |
| Performance measure bar .....                                | 140 |
| Comparative measure symbol .....                             | 141 |
| Multiple performance measures comparison .....               | 141 |
| Qualitative Range.....                                       | 142 |
| User Interaction .....                                       | 143 |
| Animation .....  | 143 |
| Responsiveness during browser resize .....                   | 143 |
| Applying same color to all ticks and labels in a range ..... | 143 |
| Tooltip.....   | 144 |
| Methods.....   | 145 |
| destroy() .....  | 145 |
| redraw() .....   | 145 |
| setComparativeMeasureSymbol(index, measure) .....            | 146 |
| setFeatureMeasureBarValue(index, measure) .....              | 146 |
| Events .....   | 147 |

|   |     |
|---|-----|
| drawCaption .....                         | 147 |
| drawCategory.....                         | 147 |
| drawComparativeMeasureSymbol.....         | 147 |
| drawFeatureMeasureBar .....               | 147 |
| drawIndicator.....                        | 148 |
| drawLabels .....                          | 148 |
| drawTicks .....                           | 148 |
| drawQualitativeRanges .....               | 149 |
| load .....                                | 149 |
| Button .....                              | 149 |
| Overview .....                            | 149 |
| Key Features .....                        | 149 |
| Create a simple Button in TypeScript..... | 150 |
| Configuring Button Properties .....       | 150 |
| Easy Customization .....                  | 151 |
| Button Size .....                         | 151 |
| Content Type .....                        | 153 |
| Prefix and Suffix icons .....             | 153 |
| Image Position .....                      | 156 |
| Theme support.....                        | 159 |
| Custom CSS .....                          | 159 |
| Button Type .....                         | 162 |
| Repeat Button.....                        | 163 |
| Custom Buttons .....                      | 164 |
| Image in Button .....                     | 167 |
| RTL Support.....                          | 167 |
| Icons.....                                | 168 |
| Adding icon in Button .....               | 168 |
| List of Icons .....                       | 169 |
| Miscellaneous .....                       | 174 |
| Text .....                                | 174 |
| Show Rounded Corner .....                 | 175 |
| Chart .....                               | 175 |
| Overview .....                            | 175 |
| Getting Started .....                     | 176 |

|   |     |
|---|-----|
| Adding Script Reference.....                      | 176 |
| Create your chart .....                           | 176 |
| Populate chart with data .....                    | 178 |
| Add Data Labels .....                             | 180 |
| Enable Legend.....                                | 182 |
| Enable Tooltip .....                              | 183 |
| Add Chart Title .....                             | 184 |
| Working with Data .....                           | 185 |
| Local Data .....                                  | 185 |
| Remote Data .....                                 | 187 |
| Chart Dimensions.....                             | 188 |
| Set size for the container .....                  | 188 |
| Set size in pixels .....                          | 189 |
| Setting size relative to the container size ..... | 189 |
| Responsive chart.....                             | 190 |
| Axis .....  | 190 |
| Category Axis .....                               | 191 |
| Numeric Axis .....                                | 196 |
| DateTime Axis .....                               | 204 |
| DateTime Category Axis .....                      | 211 |
| Logarithmic Axis .....                            | 214 |
| Label Format .....                                | 218 |
| Common axis features .....                        | 222 |
| Multiple Axis .....                               | 243 |
| Smart Axis Labels .....                           | 245 |
| Multi-level Labels .....                          | 253 |
| Multiple panes .....                              | 258 |
| Column Definitions .....                          | 261 |
| ChartTypes .....                                  | 263 |
| Line Chart.....                                   | 263 |
| Step Line Chart.....                              | 269 |
| Area Chart .....                                  | 274 |
| Range Area Chart .....                            | 275 |
| Step Area Chart.....                              | 276 |
| Spline Area Chart .....                           | 277 |

|                                 |     |
|---------------------------------|-----|
| Stacked Area Chart.....         | 278 |
| 100% Stacked Area Chart.....    | 279 |
| Column Chart .....              | 280 |
| RangeColumn Chart .....         | 286 |
| Stacked Column Chart.....       | 288 |
| 100% Stacked Column Chart ..... | 291 |
| Bar Chart .....                 | 294 |
| Stacked Bar Chart.....          | 296 |
| 100% Stacked Bar Chart .....    | 299 |
| Spline Chart.....               | 302 |
| Pie Chart .....                 | 305 |
| Doughnut Chart .....            | 311 |
| Multiple Pie Chart .....        | 318 |
| Pyramid Chart .....             | 322 |
| Funnel Chart .....              | 326 |
| Bubble Chart .....              | 329 |
| Scatter .....                   | 330 |
| HiLoOpenClose Chart .....       | 332 |
| Candle .....                    | 335 |
| HiLo .....                      | 337 |
| Polar.....                      | 338 |
| Radar Chart .....               | 341 |
| Waterfall Chart .....           | 344 |
| Error bar Chart .....           | 347 |
| Box and Whisker Chart.....      | 352 |
| Pie Of Pie Chart .....          | 355 |
| Chart Series .....              | 359 |
| Multiple Series .....           | 359 |
| Combination Series .....        | 362 |
| Data Markers .....              | 364 |
| Add Shapes .....                | 364 |
| Add image as marker .....       | 365 |
| Add labels .....                | 366 |
| Customize specific points.....  | 375 |
| Connect Line .....              | 378 |

|  |     |
|--|-----|
| Smart labels .....                                 | 379 |
| Legend .....                                       | 380 |
| Legend Visibility .....                            | 380 |
| Legend title .....                                 | 381 |
| Position and Align the Legend .....                | 382 |
| Arrange legend items in the rows and columns ..... | 384 |
| Customization .....                                | 385 |
| Handle the legend item clicked .....               | 395 |
| Series selection on legend item click .....        | 395 |
| Collapsing legend item .....                       | 396 |
| Empty Points .....                                 | 397 |
| EmptyPointSettings .....                           | 398 |
| Customizing Styles .....                           | 400 |
| Chart Title & Subtitle .....                       | 401 |
| Title .....  | 401 |
| Add Subtitle to the chart .....                    | 404 |
| Striplines .....                                   | 407 |
| Horizontal Stripline .....                         | 407 |
| Vertical Stripline .....                           | 408 |
| Customize the Text .....                           | 409 |
| Customize the Stripline .....                      | 411 |
| Change the Z-order of the stripline .....          | 412 |
| User Interactions .....                            | 413 |
| Tooltip .....                                      | 413 |
| Zooming and Panning .....                          | 418 |
| Crosshair .....                                    | 422 |
| Trackball .....                                    | 424 |
| Highlight .....                                    | 429 |
| Selection .....                                    | 435 |
| Data Editing .....                                 | 445 |
| Performance .....                                  | 447 |
| Lazy Loading .....                                 | 448 |
| Trendlines .....                                   | 448 |
| Customize the trendline styles .....               | 449 |
| Types of Trendline .....                           | 450 |

|   |     |
|---|-----|
| Forecasting.....                        | 456 |
| Trendlines Legend .....                 | 458 |
| Technical Indicators .....              | 459 |
| Bind data to render the indicator ..... | 459 |
| Indicator Types.....                    | 460 |
| Enable Tooltip .....                    | 471 |
| Annotations .....                       | 472 |
| Rotate the annotation template .....    | 473 |
| Positioning Annotation .....            | 474 |
| Annotation alignments .....             | 475 |
| Appearance .....                        | 476 |
| Custom Color Palette .....              | 476 |
| Built-in Themes .....                   | 477 |
| Point level customization .....         | 479 |
| Series border customization .....       | 480 |
| Chart area customization .....          | 480 |
| Rendering Modes.....                    | 486 |
| VML.....                                | 486 |
| SVG .....                               | 486 |
| Canvas.....                             | 486 |
| Real-Time Chart .....                   | 487 |
| Printing Chart.....                     | 488 |
| Printing Multiple chart .....           | 489 |
| Page Setup .....                        | 490 |
| 3D Chart .....                          | 490 |
| 3D Column Chart.....                    | 490 |
| 3D Bar Chart.....                       | 491 |
| 3D Stacked Column Chart .....           | 492 |
| 3D 100% Stacked Column Chart.....       | 493 |
| 3D Stacked Bar Chart .....              | 494 |
| 3D 100% Stacked Bar Chart.....          | 495 |
| 3D Pie Chart .....                      | 496 |
| 3D Doughnut Chart .....                 | 497 |
| Configure 3D Chart .....                | 498 |
| Localization .....                      | 504 |

|                                 |     |
|---------------------------------|-----|
| Public Methods .....            | 505 |
| Checkbox .....                  | 507 |
| Overview .....                  | 507 |
| Key Features .....              | 507 |
| Getting Started .....           | 507 |
| Create your first CheckBox..... | 508 |
| Easy customization .....        | 510 |
| Checked status .....            | 510 |
| Enable Tri-State.....           | 511 |
| Check State .....               | 512 |
| Checkbox Size .....             | 513 |
| Text .....                      | 514 |
| Rounded corner .....            | 515 |
| Miscellaneous .....             | 516 |
| Checkbox Id.....                | 516 |
| Checkbox Id Prefix.....         | 516 |
| Checkbox Name .....             | 516 |
| Checkbox Value.....             | 516 |
| CircularGauge .....             | 517 |
| Overview .....                  | 517 |
| Getting Started .....           | 517 |
| Create a Circular Gauge .....   | 518 |
| Set Height and Width.....       | 519 |
| Set Background Color.....       | 520 |
| Provide Scale Values .....      | 521 |
| Add Label Customization .....   | 522 |
| Add Pointers .....              | 523 |
| Add Tick Details.....           | 524 |
| Add Range Values .....          | 525 |
| Add Indicator Details.....      | 527 |
| Add Custom Label Details .....  | 529 |
| Interaction and Animation ..... | 530 |
| Scales .....                    | 535 |
| Adding Scale Collection .....   | 535 |
| Scale Customization .....       | 536 |



|                                    |     |
|------------------------------------|-----|
| Multiple Scales .....              | 540 |
| Pointers .....                     | 541 |
| Adding Pointer Collection .....    | 541 |
| Adding Pointer Value .....         | 542 |
| Pointer Styles .....               | 543 |
| Pointer Images .....               | 547 |
| Multiple Pointers .....            | 552 |
| Pointer Value Text.....            | 553 |
| Appearance .....                   | 554 |
| Font Options .....                 | 556 |
| Labels .....                       | 557 |
| Adding Label Collection.....       | 557 |
| Label Customization.....           | 558 |
| Multiple Labels.....               | 561 |
| Ticks .....                        | 562 |
| Adding Tick Collection .....       | 562 |
| Tick Customization .....           | 563 |
| Indicators .....                   | 564 |
| Adding Indicator Collection.....   | 565 |
| Basic Customization .....          | 566 |
| State Ranges .....                 | 567 |
| Multiple Indicators.....           | 568 |
| Ranges and Frames .....            | 570 |
| Adding Range Collection .....      | 570 |
| Range Customization .....          | 571 |
| Colors and Border .....            | 572 |
| Positioning the ranges .....       | 573 |
| Multiple Ranges .....              | 574 |
| Frames .....                       | 575 |
| Legend .....                       | 576 |
| Legend Visibility .....            | 576 |
| Position and Align the Legend..... | 578 |
| Customization .....                | 579 |
| Events .....                       | 585 |
| Custom labels.....                 | 586 |

|   |     |
|---|-----|
| Adding Custom Label Collection .....        | 586 |
| Multiple Custom Labels.....                 | 588 |
| Outer Custom Label .....                    | 589 |
| Tooltip.....                                | 590 |
| Default Tooltip .....                       | 590 |
| Tooltip Template.....                       | 592 |
| Sub Gauges .....                            | 593 |
| Adding SubGauges .....                      | 593 |
| Multiple SubGauges .....                    | 595 |
| Gauge Position .....                        | 597 |
| Exporting.....                              | 599 |
| Methods and Events .....                    | 600 |
| Public Methods .....                        | 600 |
| Events .....                                | 626 |
| MVVM.....                                   | 630 |
| AngularJS.....                              | 630 |
| Rendering the Circular Gauge .....          | 630 |
| Adding Scale Collection .....               | 631 |
| Adding Pointer Collection .....             | 632 |
| Adding Label Collection.....                | 633 |
| Adding Tick Collection .....                | 634 |
| Adding Range Collection .....               | 635 |
| Two Way Binding .....                       | 636 |
| ColorPicker.....                            | 638 |
| Overview .....                              | 638 |
| Getting Started .....                       | 638 |
| Creating an ColorPicker in TypeScript ..... | 638 |
| Behavior Settings .....                     | 639 |
| showPreview.....                            | 639 |
| showRecentColors .....                      | 640 |
| enableOpacity .....                         | 641 |
| columns.....                                | 642 |
| Configure Values .....                      | 643 |
| opacityValue .....                          | 643 |
| button and tooltipText.....                 | 644 |

|   |     |
|---|-----|
| Appearance and Styling .....                    | 646 |
| modelType .....                                 | 646 |
| palette.....                                    | 647 |
| basicpalette.....                               | 648 |
| custompalette .....                             | 649 |
| displayInline .....                             | 650 |
| Theme Support .....                             | 651 |
| CustomCss.....                                  | 652 |
| Keyboard Interaction .....                      | 654 |
| Configure Keyboard Interaction .....            | 654 |
| Miscellaneous .....                             | 655 |
| getValue .....                                  | 655 |
| setValue .....                                  | 655 |
| getColor .....                                  | 656 |
| CurrencyTextbox.....                            | 656 |
| Overview .....                                  | 656 |
| Getting Started .....                           | 656 |
| Creating an CurrencyTextbox in TypeScript ..... | 657 |
| set min and max values.....                     | 658 |
| Behavior Settings .....                         | 658 |
| Decimal Places .....                            | 658 |
| Persistence Support .....                       | 659 |
| Strict Mode Support.....                        | 660 |
| Enabled or Disabled .....                       | 660 |
| Adjusting CurrencyTextBox Size .....            | 661 |
| Increment Step .....                            | 662 |
| Define Name .....                               | 663 |
| Define Value.....                               | 663 |
| Define maxValue and minValue .....              | 664 |
| Read Only Support .....                         | 665 |
| Appearance.....                                 | 665 |
| Rounded Corner Support .....                    | 667 |
| Spin Button Support.....                        | 667 |
| Water Mark Text Support .....                   | 668 |
| Globalization Support .....                     | 669 |

|  |     |
|--|-----|
| Configure Globalization.....               | 669 |
| RTL Support.....                           | 670 |
| Enable RTL.....                            | 670 |
| Keyboard Interaction .....                 | 671 |
| Configuring Keyboard Navigation .....      | 671 |
| DatePicker.....                            | 672 |
| OverView .....                             | 672 |
| Key features .....                         | 672 |
| Getting Started .....                      | 672 |
| Creating an DatePicker in TypeScript ..... | 672 |
| Configuring the DatePicker properties..... | 673 |
| Behavior Settings .....                    | 674 |
| Selected Date .....                        | 674 |
| Date Range.....                            | 674 |
| Start and Depth Level.....                 | 675 |
| Display Inline Mode .....                  | 675 |
| Strict Mode .....                          | 676 |
| Formatting .....                           | 676 |
| Date Format.....                           | 677 |
| Header Format.....                         | 678 |
| Day Header .....                           | 678 |
| Tooltip with Formatting .....              | 678 |
| Globalization.....                         | 679 |
| Watermark and Today Button Text.....       | 680 |
| Accessibility.....                         | 680 |
| Keyboard Interaction .....                 | 680 |
| Customization .....                        | 681 |
| Set Dimension .....                        | 681 |
| Show Footer.....                           | 682 |
| Show Popup Button .....                    | 682 |
| Show Other Months.....                     | 682 |
| Highlight Special Date .....               | 682 |
| Validation.....                            | 683 |
| State Persistence .....                    | 684 |
| Miscellaneous .....                        | 685 |

|   |     |
|---|-----|
| Start Day .....   | 685 |
| Step Months.....  | 685 |
| Read Only.....  | 685 |
| Enable or Disable .....   | 686 |
| How to? .....   | 686 |
| Customizing template with range selection between two DatePicker..... | 686 |
| Localize DatePicker with browser specific culture .....               | 686 |
| Disable specific dates to restrict user .....                         | 686 |
| How to integrate with bootstrap grid system? .....                    | 687 |
| How to use DatePicker in AngularJS? .....                             | 687 |
| DateRangePicker.....  | 687 |
| Overview .....  | 687 |
| Key Features .....  | 687 |
| Getting Started .....   | 687 |
| Create your first DateRangePicker .....                               | 687 |
| Get/Set Value.....  | 688 |
| Behavior Settings .....   | 689 |
| Display Format .....  | 692 |
| Globalization .....   | 693 |
| Customization .....   | 693 |
| State Persistence.....  | 694 |
| Miscellaneous .....   | 695 |
| DateTimePicker.....   | 696 |
| Overview .....  | 696 |
| Key Features .....  | 696 |
| Getting Started .....   | 696 |
| Create your first DateTimePicker .....                                | 696 |
| Display format.....   | 698 |
| DateTime format.....  | 698 |
| Day Header format .....   | 700 |
| Date Range.....   | 701 |
| Start and Depth level .....   | 702 |
| Start Level .....   | 702 |
| Depth Level .....   | 703 |
| Date in other months.....   | 704 |

|   |     |
|---|-----|
| Time Interval .....                                 | 705 |
| Globalization .....                                 | 706 |
| Right-to-Left .....                                 | 708 |
| Appearance and Styling .....                        | 709 |
| Theme .....   | 709 |
| CSS Class .....                                     | 711 |
| Diagram .....                                       | 713 |
| Overview .....                                      | 713 |
| Getting started .....                               | 714 |
| Adding Script Reference .....                       | 715 |
| Initialize Diagram .....                            | 715 |
| Populate Diagram with nodes and connectors .....    | 716 |
| Dialog .....  | 717 |
| Overview .....                                      | 717 |
| Key Features .....                                  | 717 |
| Getting Started .....                               | 717 |
| Create a Dialog .....                               | 718 |
| Add dialog content .....                            | 719 |
| Set the title .....                                 | 719 |
| Open Dialog dynamically .....                       | 720 |
| Load content .....                                  | 721 |
| Action Buttons .....                                | 722 |
| Customizing Action Buttons .....                    | 723 |
| Animation .....                                     | 724 |
| Keyboard interaction .....                          | 724 |
| How To? .....                                       | 725 |
| Create Multiple Dialogs .....                       | 725 |
| Create Nested Dialog .....                          | 726 |
| Create Confirmation Dialog with Footer option ..... | 727 |
| DigitalGauge .....                                  | 729 |
| Overview .....                                      | 729 |
| Getting Started .....                               | 729 |
| Create a Digital Gauge .....                        | 730 |
| Set Height and Width values .....                   | 731 |
| Set Items Property .....                            | 731 |

|   |     |
|---|-----|
| Add Background Image .....                | 732 |
| Add Location .....                        | 733 |
| Add Items Collection .....                | 733 |
| Basic Settings .....                      | 735 |
| Height and Width Customization .....      | 735 |
| Responsive Layout .....                   | 735 |
| Themes .....                              | 736 |
| Frames .....                              | 736 |
| Inner and Outer Width Customization ..... | 736 |
| Setting Background Image .....            | 737 |
| Digital Elements .....                    | 738 |
| Segment Settings .....                    | 739 |
| Appearance .....                          | 739 |
| Dimension Modification.....               | 740 |
| Character Settings.....                   | 741 |
| Appearance.....                           | 741 |
| Count and Type.....                       | 741 |
| Text Positioning .....                    | 742 |
| Shadow Effects.....                       | 743 |
| Font Customization .....                  | 744 |
| Multiple Items.....                       | 745 |
| Exporting the Digital Gauge .....         | 746 |
| MVVM.....                                 | 747 |
| Rendering the Digital Gauge .....         | 748 |
| Adding the Digital Gauge Items.....       | 748 |
| Two Way Binding .....                     | 748 |
| Methods.....                              | 749 |
| Events .....                              | 752 |
| DropDownList .....                        | 753 |
| DropDownList .....                        | 753 |
| Key Features .....                        | 753 |
| Getting Started .....                     | 754 |
| Creating DropDownList in Typescript.....  | 754 |
| Populating data .....                     | 755 |
| Setting Dimensions .....                  | 756 |

|  |     |
|--|-----|
| Setting and Getting Value .....        | 757 |
| Rendering Mode .....                   | 757 |
| Using an input element .....           | 758 |
| Using Select Element .....             | 758 |
| Using UL-LI .....                      | 758 |
| Data Binding.....                      | 760 |
| Fields.....                            | 760 |
| Local Data .....                       | 761 |
| Binding Remote Data Service .....      | 763 |
| Virtual Scrolling.....                 | 764 |
| Checkbox .....                         | 765 |
| Selection Modes .....                  | 767 |
| Check/Uncheck All .....                | 770 |
| Functionalities.....                   | 771 |
| Selection .....                        | 771 |
| Grouping .....                         | 776 |
| Sorting.....                           | 780 |
| Cascading .....                        | 781 |
| Search .....                           | 787 |
| Template Support .....                 | 789 |
| Header Template .....                  | 789 |
| Template Field .....                   | 789 |
| Customization .....                    | 791 |
| Adding watermark text .....            | 791 |
| Applying Rounded Corner .....          | 792 |
| Enable/Disable the widget .....        | 793 |
| Applying HTML Attributes .....         | 794 |
| Setting dimensions.....                | 795 |
| Widget Sizing .....                    | 795 |
| Popup resizing.....                    | 797 |
| State Persistence .....                | 798 |
| Accessing currently stored state ..... | 799 |
| Localization .....                     | 800 |
| Load On Demand .....                   | 800 |
| Accessibility.....                     | 801 |



|  |     |
|--|-----|
| Keyboard Navigation.....   | 801 |
| How To.....  | 804 |
| Set focus to control initially?.....   | 804 |
| Clear the text of DropDownList input?.....   | 804 |
| Add an item dynamically to the DropDownList? .....   | 804 |
| Disable/ Enable the DropDownList widget? .....   | 805 |
| Control the popup visibility via methods in script showPopup ()/hidePopup ()? .....  | 805 |
| Retrieve the selected item data from select event via arguments? .....   | 805 |
| Append custom HTML in DropDownList popup outside the scroller part? .....  | 806 |
| Add check all option in popup list? .....  | 806 |
| To remove the items from DropDownList? .....   | 807 |
| Select the image rather than the text from the DropDownList when the template concept is used? .....                                     | 808 |
| Apply HTML Attributes such as color and class directly to the input element rather than the outer wrapper element of DropDownList? ..... | 810 |
| Add tooltip on hovering the DropDownList's items? .....  | 811 |
| Stop/Prevent the events (change/select) in the DropDownList? .....   | 812 |
| How can I add items in ejDropDownList at the first place in list? .....  | 812 |
| Can a DropDownList have delimiters in their JSON data source? .....  | 813 |
| FileExplorer .....   | 815 |
| Overview .....   | 815 |
| Key features .....   | 815 |
| Getting Started .....  | 815 |
| Creating an FileExplorer in TypeScript .....   | 815 |
| Resizing .....   | 817 |
| Localization .....   | 818 |
| Change the Culture .....   | 819 |
| Behavior Settings .....  | 821 |
| File type restriction .....  | 822 |
| Customize the AJAX request settings .....  | 822 |
| User Interface .....   | 824 |
| File Actions.....  | 824 |
| File action through Toolbar .....  | 824 |
| File action through Context menu .....   | 825 |
| Toolbar.....   | 825 |

|  |     |
|--|-----|
| Toolbar items .....                                  | 825 |
| Toolbar Visibility .....                             | 827 |
| Toolbar Configuration .....                          | 827 |
| Search bar .....                                     | 828 |
| Custom Tool in Toolbar .....                         | 829 |
| Enable / Disable the Toolbar Item.....               | 830 |
| Customizing the Upload Functionality .....           | 830 |
| Context Menu .....                                   | 831 |
| Context menu items.....                              | 831 |
| Context menu Visibility .....                        | 833 |
| Enable / Disable the Context menu Item .....         | 833 |
| Context Menu Events.....                             | 834 |
| Customization .....                                  | 835 |
| Dimension Customization .....                        | 835 |
| Customizing the Navigation pane .....                | 835 |
| Customizing the Content pane.....                    | 835 |
| Footer Customization.....                            | 837 |
| Customize the Root Folder name in FileExplorer ..... | 838 |
| Multiple Selection .....                             | 838 |
| Resizing .....                                       | 839 |
| Responsiveness .....                                 | 839 |
| Restriction on Resize .....                          | 840 |
| Drag and Drop Support .....                          | 840 |
| Thumbnail Compression .....                          | 841 |
| Accessing shared folder .....                        | 841 |
| Localization .....                                   | 842 |
| Change the Culture .....                             | 844 |
| Keyboard Navigation.....                             | 846 |
| Gantt.....   | 848 |
| Overview .....                                       | 848 |
| Getting Started .....                                | 848 |
| Create your first Gantt in TypeScript .....          | 848 |
| Adding script references .....                       | 849 |
| Initialize the Gantt .....                           | 849 |
| Enable Toolbar .....                                 | 851 |

|                                     |     |
|-------------------------------------|-----|
| Enable Sorting .....                | 852 |
| Enable Editing .....                | 852 |
| Enable Context Menu .....           | 854 |
| Enable Column Menu.....             | 854 |
| Provide tasks relationship .....    | 855 |
| Provide Resources.....              | 856 |
| Highlight Weekend.....              | 857 |
| Grid .....                          | 858 |
| Overview .....                      | 858 |
| Getting started .....               | 859 |
| Script and CSS Reference .....      | 859 |
| Create a Grid .....                 | 861 |
| Data Binding.....                   | 862 |
| Enable Paging.....                  | 863 |
| Enable Filtering .....              | 864 |
| Enable Grouping .....               | 865 |
| Add Summaries .....                 | 867 |
| Data binding.....                   | 869 |
| Local Data .....                    | 869 |
| Remote Data .....                   | 871 |
| Columns .....                       | 872 |
| Column Template.....                | 873 |
| Command Column .....                | 874 |
| Column Chooser.....                 | 878 |
| Foreign Key Column .....            | 879 |
| Row .....                           | 881 |
| Details Template .....              | 881 |
| Row Template .....                  | 883 |
| Drag-and-Drop .....                 | 885 |
| Editing.....                        | 891 |
| Toolbar with edit option .....      | 892 |
| Cell edit type and its params ..... | 893 |
| Cell Edit Template .....            | 895 |
| Edit Modes .....                    | 897 |
| Confirmation messages.....          | 917 |

|  |     |
|--|-----|
| Column Validation.....   | 920 |
| Persisting data in Server.....                                 | 925 |
| Adding New Row Position .....                                  | 933 |
| Render with blank row for easy add new.....                    | 934 |
| Default column values on add new .....                         | 936 |
| Filtering .....  | 938 |
| Menu filter .....  | 939 |
| Excel-like filter.....   | 942 |
| Filter bar.....  | 946 |
| Filter Operators.....  | 949 |
| FilterBar Template .....                                       | 950 |
| Selection .....  | 952 |
| Types of Selection .....                                       | 952 |
| Row Selection .....  | 953 |
| Multiple Row Selection using Checkbox Column .....             | 954 |
| Cell Selection.....  | 955 |
| Column Selection .....   | 957 |
| Touch options .....  | 959 |
| Toggle Selection .....   | 960 |
| GroupButton .....  | 961 |
| Overview .....   | 961 |
| Key Features .....   | 961 |
| Creating a GroupButton in TypeScript .....                     | 961 |
| Configuring APIs in GroupButton .....                          | 962 |
| Behavior Settings .....  | 962 |
| Different modes of button .....                                | 962 |
| SelectedItemIndex .....  | 964 |
| Select and deselect the button items using public method ..... | 965 |
| Customization .....  | 966 |
| Size.....  | 966 |
| Set Dimension .....  | 967 |
| ContentType .....  | 967 |
| Icons.....   | 968 |
| Orientation.....   | 968 |
| DataSource .....   | 969 |

|                                     |     |
|-------------------------------------|-----|
| Local Data .....                    | 969 |
| Remote Data .....                   | 970 |
| Miscellaneous .....                 | 971 |
| Show/Hide the items .....           | 971 |
| Enable/Disable .....                | 971 |
| Getting Index of given Element..... | 972 |
| Getting state of given Button ..... | 972 |
| HeatMap .....                       | 973 |
| Overview .....                      | 973 |
| Key features: .....                 | 973 |
| Getting Started .....               | 973 |
| Initialize HeatMap .....            | 973 |
| Initialize Legend .....             | 974 |
| Kanban Board .....                  | 975 |
| Overview .....                      | 975 |
| Getting Started .....               | 975 |
| Preparing HTML document .....       | 976 |
| Adding Script Reference.....        | 976 |
| Create a Kanban.....                | 977 |
| Data Binding.....                   | 977 |
| Mapping Values .....                | 978 |
| Enable Swimlane .....               | 979 |
| Adding Filters .....                | 980 |
| Columns .....                       | 981 |
| Key Mapping .....                   | 981 |
| Multiple Key Mapping.....           | 983 |
| Headers.....                        | 984 |
| Width .....                         | 985 |
| Visibility.....                     | 986 |
| Toggle .....                        | 987 |
| Allow Dragging .....                | 988 |
| Allow Dropping .....                | 989 |
| Items Count.....                    | 990 |
| Customize Items Count Text .....    | 991 |
| Data Binding.....                   | 992 |

|  |      |
|--|------|
| Local Data .....                         | 992  |
| Remote Data .....                        | 994  |
| Workflows.....                           | 995  |
| Swim lanes .....                         | 996  |
| Drag And Drop between swim lanes .....   | 997  |
| Unassigned swim lane group .....         | 999  |
| Drag and Drop.....                       | 1002 |
| Prioritization of cards.....             | 1003 |
| Cards .....                              | 1004 |
| Customization .....                      | 1004 |
| Template .....                           | 1006 |
| Tooltip.....                             | 1008 |
| Collapsible Cards .....                  | 1012 |
| Editing.....                             | 1013 |
| Configuring Edit Items.....              | 1014 |
| Edit modes .....                         | 1015 |
| Cell edit type and its params .....      | 1025 |
| Column Validation.....                   | 1027 |
| Filtering.....                           | 1029 |
| Scrolling .....                          | 1030 |
| Set width and height in pixel.....       | 1030 |
| Set height and width in percentage ..... | 1032 |
| Set width as auto .....                  | 1033 |
| Enabling freeze swim lane .....          | 1034 |
| Selection and Hovering .....             | 1035 |
| Types of Selection .....                 | 1036 |
| Responsive.....                          | 1037 |
| Mobile Layout .....                      | 1038 |
| Width .....                              | 1045 |
| Min Width.....                           | 1046 |
| Stacked Headers .....                    | 1047 |
| Adding Stacked header columns.....       | 1047 |
| Context Menu .....                       | 1048 |
| Default Context Menu items .....         | 1049 |
| Custom Context Menu .....                | 1050 |

|  |      |
|--|------|
| Sub Context Menu .....                       | 1052 |
| Print .....                                  | 1054 |
| Localization .....                           | 1055 |
| Localization .....                           | 1055 |
| Right to Left (RTL) .....                    | 1058 |
| Styling .....                                | 1059 |
| List of classes and its purposes .....       | 1059 |
| Web Accessibility .....                      | 1060 |
| Keyboard Navigation.....                     | 1060 |
| LinearGauge.....                             | 1061 |
| Overview .....                               | 1061 |
| Getting Started .....                        | 1061 |
| Create a Linear Gauge .....                  | 1062 |
| Set Height and Width values .....            | 1064 |
| Set Animation option and Label Color .....   | 1064 |
| Provide Scale Values .....                   | 1065 |
| Add Pointers .....                           | 1066 |
| Add Label Customization .....                | 1067 |
| Add Tick Details.....                        | 1068 |
| Add Custom Label Details .....               | 1069 |
| Change Scale from Degree to Fahrenheit ..... | 1070 |
| Add Custom Label for Current Value.....      | 1071 |
| Basic Settings .....                         | 1073 |
| Adding Dimension.....                        | 1073 |
| Adding frame .....                           | 1074 |
| Appearance.....                              | 1076 |
| Responsive .....                             | 1077 |
| Localization .....                           | 1079 |
| Interaction and Animation .....              | 1079 |
| Scales .....                                 | 1081 |
| Adding scale collection.....                 | 1081 |
| Scale Customization .....                    | 1082 |
| Appearance.....                              | 1084 |
| Scale Types.....                             | 1086 |
| Adding multiple scales .....                 | 1090 |

|  |      |
|--|------|
| Marker Pointers .....                  | 1092 |
| Adding marker pointer collection ..... | 1092 |
| Add marker pointer value .....         | 1093 |
| Pointer Styles .....                   | 1095 |
| Positioning the pointer .....          | 1096 |
| Types.....                             | 1097 |
| Bar Pointers .....                     | 1099 |
| Adding bar pointer collection.....     | 1099 |
| Adding bar pointer value .....         | 1101 |
| Pointer Styles .....                   | 1102 |
| Positioning the pointer .....          | 1104 |
| Multiple Bar Pointers .....            | 1105 |
| Labels .....                           | 1107 |
| Label Customization.....               | 1107 |
| Unit text and Positioning .....        | 1109 |
| Ticks .....                            | 1111 |
| Adding tick collection .....           | 1111 |
| Tick Customization .....               | 1112 |
| Types.....                             | 1114 |
| Positioning the ticks .....            | 1115 |
| Ranges .....                           | 1117 |
| Adding range collection .....          | 1117 |
| Range Customization .....              | 1118 |
| Colors and Border .....                | 1120 |
| Positioning the ranges .....           | 1121 |
| Multiple Ranges .....                  | 1123 |
| Custom labels.....                     | 1124 |
| Adding Custom label collection.....    | 1125 |
| Basic Customization .....              | 1126 |
| Locating the CustomLabels .....        | 1128 |
| Multiple Custom Labels.....            | 1130 |
| Indicators .....                       | 1132 |
| Setting Dimension.....                 | 1132 |
| State Ranges .....                     | 1134 |
| Color and Appearance .....             | 1136 |



|  |      |
|--|------|
| Font options .....                     | 1138 |
| Multiple Indicator .....               | 1140 |
| Exporting .....                        | 1142 |
| MVVM .....                             | 1144 |
| Rendering the Linear gauge .....       | 1144 |
| Adding Scale collection .....          | 1145 |
| Adding Marker Pointer collection ..... | 1146 |
| Adding label collection .....          | 1147 |
| Adding Tick collection .....           | 1148 |
| Adding Range collection .....          | 1149 |
| Two Way Binding .....                  | 1150 |
| Events .....                           | 1151 |
| Methods .....                          | 1155 |
| ListBox .....                          | 1187 |
| Overview .....                         | 1187 |
| Key Features .....                     | 1187 |
| Getting Started .....                  | 1187 |
| Create a ListBox .....                 | 1187 |
| Data Binding .....                     | 1189 |
| Selection .....                        | 1190 |
| Data binding .....                     | 1190 |
| Field mapping .....                    | 1190 |
| Local data .....                       | 1190 |
| Remote data .....                      | 1191 |
| Selection .....                        | 1196 |
| Selection on initialize .....          | 1196 |
| Multiple selection .....               | 1197 |
| Checkbox .....                         | 1198 |
| Sorting .....                          | 1198 |
| Grouping .....                         | 1200 |
| Using span tag .....                   | 1200 |
| Data binding .....                     | 1201 |
| Templates .....                        | 1202 |
| Cascading .....                        | 1203 |
| Multilevel cascading .....             | 1205 |

|  |      |
|--|------|
| Drag and drop .....                                  | 1207 |
| Transferring a ListBox data to another ListBox ..... | 1207 |
| Dynamically set data source on drag and drop .....   | 1209 |
| Reordering .....                                     | 1211 |
| Keyboard interaction .....                           | 1212 |
| Incremental Search .....                             | 1212 |
| ListView.....  | 1214 |
| Overview .....                                       | 1214 |
| Key Features .....                                   | 1214 |
| Getting Started .....                                | 1214 |
| Create a ListView .....                              | 1214 |
| Add Header .....                                     | 1216 |
| Grouped List.....                                    | 1216 |
| Selection .....                                      | 1219 |
| Customize Header.....                                | 1222 |
| Data Binding.....                                    | 1224 |
| Local Data Binding.....                              | 1224 |
| Remote Data Binding .....                            | 1225 |
| FieldSettings.....                                   | 1228 |
| Filtering.....                                       | 1230 |
| Dimensions .....                                     | 1231 |
| Virtual Scrolling.....                               | 1233 |
| Normal Mode.....                                     | 1233 |
| Continuous Mode .....                                | 1233 |
| Maps .....   | 1233 |
| Maps .....   | 1233 |
| Use Case Scenarios .....                             | 1234 |
| Key Features .....                                   | 1234 |
| Getting Started .....                                | 1234 |
| Add Libraries .....                                  | 1235 |
| Initialize Map .....                                 | 1237 |
| Data Binding in Map.....                             | 1239 |
| Enable Tooltip .....                                 | 1243 |
| Legend .....   | 1244 |
| Populate Data .....                                  | 1246 |

|  |      |
|--|------|
| Shape Data .....                         | 1246 |
| Data Binding.....                        | 1246 |
| Customization .....                      | 1249 |
| Shape Settings.....                      | 1249 |
| Color Mapping .....                      | 1250 |
| Color Palette .....                      | 1254 |
| Tooltip.....                             | 1256 |
| Map Elements.....                        | 1260 |
| Markers.....                             | 1260 |
| Bubbles .....                            | 1263 |
| Legend .....                             | 1265 |
| User Interaction .....                   | 1270 |
| Map Selection .....                      | 1270 |
| MultiSelection.....                      | 1271 |
| Dragging On Selection.....               | 1272 |
| Zooming .....                            | 1273 |
| Panning .....                            | 1276 |
| Navigation Control .....                 | 1276 |
| Layers.....                              | 1279 |
| Multilayer.....                          | 1279 |
| SubLayer .....                           | 1279 |
| Map Providers.....                       | 1280 |
| Open Street Map .....                    | 1280 |
| Bing Map.....                            | 1281 |
| AngularJS Support.....                   | 1282 |
| Methods.....                             | 1284 |
| Events .....                             | 1286 |
| MaskEdit .....                           | 1289 |
| Overview.....                            | 1289 |
| Getting Started .....                    | 1289 |
| Creating an MaskEdit in TypeScript ..... | 1289 |
| Error Visibility.....                    | 1290 |
| CustomCharacter .....                    | 1291 |
| Behavior Settings .....                  | 1291 |
| Persistence Support .....                | 1291 |

|   |      |
|---|------|
| Enabled or Disabled .....               | 1292 |
| Adjusting MaskEdit Size .....           | 1292 |
| Define Value.....                       | 1293 |
| Read Only Support .....                 | 1294 |
| MaskEdit Properties.....                | 1294 |
| HidePromptOnLeave .....                 | 1294 |
| InputMode .....                         | 1294 |
| MaskFormat .....                        | 1294 |
| Globalization Support in MaskEdit.....  | 1295 |
| Appearance .....                        | 1297 |
| Theme .....                             | 1297 |
| CSS Class .....                         | 1297 |
| Rounded Corner Support .....            | 1298 |
| WatermarkText Support .....             | 1299 |
| Text Alignment Support .....            | 1299 |
| Menu .....                              | 1300 |
| Overview .....                          | 1300 |
| Getting Started .....                   | 1300 |
| Create a Menu .....                     | 1301 |
| Configure parent Menu items .....       | 1302 |
| Initialize sub-level Menu items .....   | 1302 |
| Define multiple level Menu items .....  | 1303 |
| Orientation .....                       | 1304 |
| Horizontal Menu .....                   | 1305 |
| Vertical Menu .....                     | 1306 |
| Data binding.....                       | 1307 |
| Field Members .....                     | 1307 |
| Local data .....                        | 1308 |
| Remote data .....                       | 1309 |
| Icons and navigation .....              | 1310 |
| Icons.....                              | 1310 |
| Navigation.....                         | 1312 |
| Customizing the Submenu direction ..... | 1314 |
| Look and feel.....                      | 1318 |
| cssClass .....                          | 1318 |

|   |      |
|---|------|
| Background Template .....                     | 1320 |
| Context Menu .....                            | 1323 |
| HideContextMenu .....                         | 1325 |
| ShowContextMenu .....                         | 1325 |
| Center Menu .....                             | 1326 |
| RTL Support.....                              | 1327 |
| Separators.....                               | 1328 |
| Separators for Context Menu .....             | 1330 |
| Responsive Layout .....                       | 1331 |
| Responsive in Desktop: .....                  | 1333 |
| Responsive in Mobile or Tablet:.....          | 1333 |
| Keyboard Navigation.....                      | 1334 |
| Miscellaneous .....                           | 1336 |
| Height .....                                  | 1336 |
| Width .....                                   | 1337 |
| Open on click .....                           | 1337 |
| Animation .....                               | 1337 |
| Title text.....                               | 1338 |
| Show root level arrows .....                  | 1339 |
| Show sub level arrows .....                   | 1339 |
| Navigation Drawer .....                       | 1340 |
| Overview .....                                | 1340 |
| Getting Started .....                         | 1340 |
| Create a Navigation Drawer .....              | 1340 |
| Animations.....                               | 1345 |
| Customize Direction.....                      | 1346 |
| Customize Position.....                       | 1348 |
| TargetId.....                                 | 1349 |
| NumericTextbox .....                          | 1351 |
| Overview .....                                | 1351 |
| Getting Started .....                         | 1351 |
| Creating an NumericTextbox in TypeScript..... | 1352 |
| set min and max values.....                   | 1353 |
| Behavior Settings .....                       | 1353 |
| Decimal Places .....                          | 1353 |

|   |      |
|---|------|
| Persistence Support .....                                 | 1354 |
| Strict Mode Support.....                                  | 1355 |
| Enabled or Disabled .....                                 | 1356 |
| Adjusting Textbox Size .....                              | 1356 |
| Increment Step .....                                      | 1357 |
| Define Name .....   | 1358 |
| Define Value.....   | 1358 |
| Define max <b>Value</b> and min <b>Value</b> .....        | 1359 |
| Read Only Support .....                                   | 1360 |
| Appearance.....   | 1361 |
| Rounded Corner Support .....                              | 1362 |
| Spin Button Support.....                                  | 1363 |
| Water Mark Text Support .....                             | 1363 |
| Globalization Support .....                               | 1364 |
| RTL Support.....  | 1365 |
| Enable RTL.....   | 1365 |
| Keyboard Interaction .....                                | 1366 |
| Configuring Keyboard Navigation .....                     | 1366 |
| How To.....   | 1367 |
| Use Normal Textboxes as Syncfusion textboxes .....        | 1367 |
| PDF viewer .....  | 1367 |
| Overview .....  | 1367 |
| Getting Started .....                                     | 1368 |
| Script and CSS Reference .....                            | 1368 |
| Percentage <b>Textbox</b> .....                           | 1370 |
| Overview .....  | 1370 |
| Getting Started .....                                     | 1370 |
| Creating an Percentage <b>Textbox</b> in TypeScript ..... | 1370 |
| set min and max values.....                               | 1371 |
| Behavior Settings .....                                   | 1372 |
| Decimal Places .....                                      | 1372 |
| Persistence Support .....                                 | 1373 |
| Strict Mode Support.....                                  | 1374 |
| Enabled or Disabled .....                                 | 1375 |
| Adjusting Percentage <b>Text</b> Box Size .....           | 1376 |

|                                       |      |
|---------------------------------------|------|
| Increment Step .....                  | 1377 |
| Define Name .....                     | 1378 |
| Define Value.....                     | 1378 |
| Define maxValu and minValu .....      | 1379 |
| Read Only Support .....               | 1380 |
| Appearance.....                       | 1381 |
| Rounded Corner Support .....          | 1383 |
| Spin Button Support.....              | 1384 |
| Water Mark Text Support .....         | 1385 |
| Globalization Support .....           | 1385 |
| Configure Globalization.....          | 1386 |
| RTL Support.....                      | 1387 |
| Enable RTL.....                       | 1387 |
| Keyboard Interaction .....            | 1388 |
| Configuring Keyboard Navigation ..... | 1388 |
| PivotChart .....                      | 1389 |
| Overview .....                        | 1389 |
| Getting Started .....                 | 1389 |
| Script and CSS Reference .....        | 1389 |
| Relational .....                      | 1390 |
| OLAP .....                            | 1393 |
| Dimensions .....                      | 1394 |
| Set size in percentage .....          | 1394 |
| Set size in pixels .....              | 1395 |
| Responsive .....                      | 1396 |
| Chart types.....                      | 1397 |
| Column chart .....                    | 1397 |
| Stacking column chart.....            | 1398 |
| Bar chart .....                       | 1399 |
| Stacking bar chart .....              | 1400 |
| Pie chart .....                       | 1401 |
| Pyramid chart.....                    | 1402 |
| Funnel chart .....                    | 1403 |
| Line chart .....                      | 1404 |
| Step line chart .....                 | 1405 |

|  |      |
|--|------|
| Spline chart .....                                 | 1406 |
| Area chart .....                                   | 1407 |
| Step area chart.....                               | 1408 |
| Spline area chart .....                            | 1409 |
| Stacking area chart.....                           | 1410 |
| Doughnut chart.....                                | 1411 |
| Scatter chart .....                                | 1412 |
| Bubble chart.....                                  | 1413 |
| Combination chart .....                            | 1413 |
| Drill operation.....                               | 1414 |
| Legend .....                                       | 1415 |
| Legend visibility.....                             | 1415 |
| Legend shape .....                                 | 1416 |
| Legend position.....                               | 1417 |
| Legend title .....                                 | 1418 |
| Legend alignment .....                             | 1419 |
| Legend items - size and border .....               | 1420 |
| Legend border.....                                 | 1421 |
| Legend text .....                                  | 1422 |
| Axes .....   | 1423 |
| Label format.....                                  | 1423 |
| Common axis features .....                         | 1425 |
| Multi-level labels.....                            | 1432 |
| Relational .....                                   | 1433 |
| OLAP .....   | 1433 |
| Multiple Axes .....                                | 1433 |
| Customizing axes at row index of zero .....        | 1434 |
| Customizing axes at row index one.....             | 1435 |
| Customizing axes at column index of zero .....     | 1436 |
| Customizing axes at column index of one .....      | 1436 |
| Customizing series .....                           | 1437 |
| Multiple axes support by series index.....         | 1438 |
| Customizing PrimaryYAxis and axes properties ..... | 1439 |
| Exporting.....                                     | 1441 |
| Excel export .....                                 | 1442 |



|                                |      |
|--------------------------------|------|
| Word export.....               | 1442 |
| PDF export .....               | 1443 |
| Image export.....              | 1444 |
| Exporting customization .....  | 1445 |
| PivotGrid.....                 | 1449 |
| Overview.....                  | 1449 |
| Getting Started .....          | 1450 |
| Script and CSS Reference ..... | 1450 |
| Relational .....               | 1450 |
| OLAP .....                     | 1453 |
| PivotGauge.....                | 1454 |
| Overview.....                  | 1454 |
| Getting Started .....          | 1455 |
| Script and CSS Reference ..... | 1455 |
| Relational .....               | 1456 |
| OLAP .....                     | 1459 |
| Frame type.....                | 1462 |
| Full circle.....               | 1462 |
| Half circle .....              | 1463 |
| Scale.....                     | 1464 |
| Adding scale .....             | 1464 |
| Scale customization .....      | 1465 |
| Pointers.....                  | 1467 |
| Pointer types.....             | 1467 |
| Adding pointer collection..... | 1469 |
| Appearance Customization ..... | 1470 |
| Pointer position .....         | 1471 |
| Pointer image.....             | 1472 |
| Pointer value text.....        | 1473 |
| Labels.....                    | 1474 |
| Adding label collection.....   | 1474 |
| Appearance customization ..... | 1475 |
| Unit text.....                 | 1476 |
| PivotTreeMap .....             | 1477 |
| Overview.....                  | 1477 |

|   |      |
|---|------|
| Getting Started .....                                     | 1477 |
| Script and CSS Reference .....                            | 1478 |
| Drill operation.....                                      | 1480 |
| Named sets .....  | 1481 |
| Color mapping.....  | 1482 |
| Legend .....  | 1485 |
| Legend visibility.....                                    | 1485 |
| ProgressBar .....   | 1486 |
| Overview .....  | 1486 |
| Getting Started .....                                     | 1486 |
| Create a ProgressBar .....                                | 1487 |
| Progress Control using Length of the Password Field ..... | 1489 |
| Define value .....  | 1492 |
| Value .....   | 1492 |
| Percentage .....  | 1493 |
| Setting Range .....                                       | 1493 |
| Appearance and Styling .....                              | 1494 |
| Adjusting ProgressBar size .....                          | 1494 |
| Custom text.....  | 1495 |
| Theme .....   | 1495 |
| CSS class.....  | 1496 |
| Enabling the ProgressBar .....                            | 1496 |
| State Maintenance.....                                    | 1497 |
| RTL Support.....  | 1498 |
| How To.....   | 1499 |
| How to increment & show the progressbar movement.....     | 1499 |
| RadialMenu.....   | 1502 |
| Overview.....   | 1502 |
| Key Features .....  | 1502 |
| Getting Started .....                                     | 1502 |
| Create a RadialMenu.....                                  | 1503 |
| Image and text configuration.....                         | 1504 |
| Displaying RadialMenu.....                                | 1505 |
| RadialMenu item functionalities .....                     | 1507 |
| Dimension.....  | 1508 |

|   |      |
|---|------|
| Item Customization .....                          | 1510 |
| Image Customization .....                         | 1517 |
| Appearance and Styling .....                      | 1520 |
| Theme .....                                       | 1520 |
| CSS Class .....                                   | 1521 |
| Template Support .....                            | 1522 |
| RadialSlider .....                                | 1526 |
| Overview .....                                    | 1526 |
| Key Features .....                                | 1526 |
| Getting Started .....                             | 1527 |
| Create a Radial Slider control .....              | 1527 |
| Dimension .....                                   | 1528 |
| Stroke Width .....                                | 1528 |
| Setting radius .....                              | 1529 |
| Behavior settings .....                           | 1530 |
| Show/hide RadialSlider on initial rendering ..... | 1530 |
| Value accuracy .....                              | 1530 |
| Display inline .....                              | 1530 |
| Modifying Label Space .....                       | 1531 |
| Show/Hide inner circle .....                      | 1532 |
| Values customization .....                        | 1533 |
| Image customization .....                         | 1534 |
| Customizing inner circle .....                    | 1534 |
| Animation Effect .....                            | 1536 |
| Display Angle Support .....                       | 1537 |
| Start Angle .....                                 | 1537 |
| End Angle .....                                   | 1538 |
| Appearance and Styling .....                      | 1539 |
| Theme .....                                       | 1539 |
| Theme .....                                       | 1539 |
| CSS Class .....                                   | 1540 |
| RadioButton .....                                 | 1541 |
| Overview .....                                    | 1541 |
| Getting Started .....                             | 1542 |
| Create your first Radio Button .....              | 1542 |

|                                  |      |
|----------------------------------|------|
| Easy Customization .....         | 1544 |
| Checked status .....             | 1544 |
| Text .....                       | 1545 |
| Size.....                        | 1546 |
| RTL Support.....                 | 1548 |
| Miscellaneous .....              | 1550 |
| RadioButton ID.....              | 1550 |
| RadioButton Prefix id .....      | 1550 |
| RadioButton Name.....            | 1551 |
| RadioButton Value .....          | 1551 |
| RangeNavigator .....             | 1551 |
| Overview .....                   | 1551 |
| Getting Started .....            | 1552 |
| Create your RangeNavigator ..... | 1552 |
| Configure ejRangeNavigator ..... | 1552 |
| Add series .....                 | 1554 |
| Enable tooltip.....              | 1555 |
| Update Chart.....                | 1555 |
| Set value type .....             | 1556 |
| Numeric Type.....                | 1557 |
| DateTime .....                   | 1558 |
| DateTime Intervals .....         | 1558 |
| User Interactions .....          | 1573 |
| Highlight.....                   | 1573 |
| Selection .....                  | 1574 |
| Scrollbar .....                  | 1575 |
| How to .....                     | 1576 |
| Rating.....                      | 1577 |
| Overview .....                   | 1577 |
| Getting Started .....            | 1577 |
| Create a Rating Widget .....     | 1577 |
| Set the Min and Max Value .....  | 1580 |
| Set Precision.....               | 1581 |
| Rating Customization .....       | 1582 |
| Setting Value .....              | 1582 |

|   |      |
|---|------|
| Set Precision.....                                | 1585 |
| Increment Step .....                              | 1586 |
| Resetting values .....                            | 1587 |
| Read only .....                                   | 1588 |
| Enable or Disable .....                           | 1589 |
| Appearance and Styling .....                      | 1590 |
| Show ToolTip .....                                | 1590 |
| Adjusting Rating Size .....                       | 1591 |
| Theme .....                                       | 1592 |
| Custom styles .....                               | 1593 |
| Orientation .....                                 | 1594 |
| ReportViewer.....                                 | 1595 |
| Overview .....                                    | 1595 |
| Key Features .....                                | 1595 |
| Getting Started .....                             | 1596 |
| Create your first ReportViewer in Typescript..... | 1596 |
| Script and CSS Reference .....                    | 1596 |
| Load SSRS Server Reports .....                    | 1598 |
| Load RDLC Reports .....                           | 1599 |
| Limitations .....                                 | 1601 |
| RDL Specification support .....                   | 1601 |
| Layout process .....                              | 1601 |
| Unsupported expression .....                      | 1601 |
| How To.....                                       | 1602 |
| Load unsupported fonts.....                       | 1602 |
| Ribbon.....                                       | 1602 |
| Overview .....                                    | 1602 |
| Key Features .....                                | 1602 |
| Getting Started .....                             | 1602 |
| Script & CSS Reference.....                       | 1603 |
| Control Initialization.....                       | 1604 |
| Adding Tabs .....                                 | 1605 |
| Configuring Groups .....                          | 1605 |
| Adding Controls to Group .....                    | 1606 |
| User Interface .....                              | 1607 |

|                                      |      |
|--------------------------------------|------|
| Ribbon Dependencies .....            | 1608 |
| Application Tab .....                | 1609 |
| Application Menu.....                | 1609 |
| Backstage Page .....                 | 1612 |
| Tab .....                            | 1614 |
| Group.....                           | 1616 |
| Adding Tab Groups.....               | 1616 |
| Group Expander .....                 | 1620 |
| Controls Support.....                | 1621 |
| Built in Controls .....              | 1622 |
| Custom.....                          | 1625 |
| Contextual Tabs .....                | 1627 |
| Gallery.....                         | 1629 |
| Gallery Items.....                   | 1629 |
| Custom Gallery Items.....            | 1632 |
| Resize.....                          | 1634 |
| Tablet Layout .....                  | 1634 |
| Mobile Layout .....                  | 1636 |
| Mobile Toolbar Customization.....    | 1638 |
| Group Button Customization .....     | 1645 |
| Screen Tips.....                     | 1649 |
| HTML Tooltip .....                   | 1649 |
| Custom Tooltip.....                  | 1650 |
| Quick Access Toolbar .....           | 1659 |
| Globalization and Localization ..... | 1662 |
| Localization .....                   | 1662 |
| Right to Left - RTL.....             | 1664 |
| Appearance and Styling .....         | 1665 |
| CssClass.....                        | 1665 |
| Themes .....                         | 1667 |
| Customize Styles .....               | 1667 |
| Load on Demand.....                  | 1668 |
| Initially Collapsible .....          | 1671 |
| How to .....                         | 1674 |
| Get Ribbon object .....              | 1674 |

|   |      |
|---|------|
| RichTextEditor.....                         | 1675 |
| Overview.....                               | 1675 |
| Getting Started .....                       | 1675 |
| Script and CSS reference.....               | 1675 |
| Control Initialization.....                 | 1676 |
| Toolbar–Configuration.....                  | 1676 |
| Setting and Getting Content .....           | 1676 |
| Toolbar Configuration.....                  | 1677 |
| Toolbar Items.....                          | 1678 |
| Rearrange Group.....                        | 1680 |
| Undo and Redo .....                         | 1681 |
| Clipboard Operations.....                   | 1682 |
| Types of responsive toolbar.....            | 1682 |
| Context Menu.....                           | 1683 |
| Working with Content.....                   | 1684 |
| Iframe Attributes .....                     | 1684 |
| Content Editable .....                      | 1684 |
| Submit Content.....                         | 1685 |
| Persistence.....                            | 1686 |
| Apply Font color and Background color ..... | 1686 |
| Insert the content at cursor .....          | 1687 |
| Working with Selection .....                | 1687 |
| Select All .....                            | 1687 |
| Select a Range.....                         | 1688 |
| Get Selection.....                          | 1688 |
| Working with Tables .....                   | 1689 |
| Create a Table .....                        | 1689 |
| Insert a Table .....                        | 1689 |
| Insert and Delete a Row or Column .....     | 1691 |
| Format a Table.....                         | 1691 |
| Delete a table.....                         | 1692 |
| Working with Hyperlinks.....                | 1693 |
| Add and Edit a hyperlink.....               | 1693 |
| Remove a hyperlink .....                    | 1693 |
| Image and File browser.....                 | 1694 |

|   |      |
|---|------|
| Insert a Image from Online Source .....                                     | 1694 |
| Insert a Image from Your Computer .....                                     | 1694 |
| Image Properties.....   | 1696 |
| Resize an Image .....   | 1696 |
| Suppression of the Image Browser .....                                      | 1697 |
| Find and Replace.....   | 1698 |
| Footer .....  | 1699 |
| Source View .....   | 1700 |
| HTML Tag Info.....  | 1700 |
| Characters Count/Word Count .....   | 1700 |
| Clear Format .....  | 1701 |
| Resize Handle.....  | 1701 |
| Characters Count/Word Count .....   | 1702 |
| Validation.....   | 1703 |
| jQuery Validation Methods.....  | 1703 |
| Validation Rules .....  | 1703 |
| Validation Messages .....   | 1703 |
| XHTML Validation .....  | 1705 |
| Zoom.....   | 1706 |
| Print .....   | 1706 |
| Working with Lists.....   | 1707 |
| Create a Lists.....   | 1707 |
| Custom Lists .....  | 1708 |
| Clean unwanted elements and styles when copy paste from Microsoft Word..... | 1711 |
| Localization .....  | 1713 |
| Keyboard Support .....  | 1719 |
| How To.....   | 1722 |
| Add Google web fonts to editor.....   | 1722 |
| Increase RTE max word count.....  | 1723 |
| Add multiple editor instances to a single page .....                        | 1723 |
| Set the horizontal scroller rather than text wrapping in the RTE? .....     | 1725 |
| Set Toolbar Height .....  | 1725 |
| Add Separator in the Toolbar.....   | 1726 |
| Custom image for the Tools .....  | 1726 |
| Rotator.....  | 1728 |



|  |      |
|--|------|
| Overview .....                           | 1728 |
| Key Features .....                       | 1728 |
| Getting Started .....                    | 1728 |
| Create a Rotator.....                    | 1729 |
| Initialize the control .....             | 1730 |
| Data Binding.....                        | 1731 |
| Data fields and Configuration .....      | 1731 |
| DataSource.....                          | 1731 |
| Fields.....                              | 1731 |
| Text .....                               | 1731 |
| URL.....                                 | 1732 |
| Query .....                              | 1732 |
| Local data binding .....                 | 1732 |
| Behavior settings .....                  | 1733 |
| Enabling rotator .....                   | 1733 |
| Responsive rotator .....                 | 1734 |
| Auto Play .....                          | 1734 |
| Stop on hover.....                       | 1735 |
| Pager settings.....                      | 1735 |
| Show options .....                       | 1737 |
| Thumbnail .....                          | 1740 |
| Orientation .....                        | 1742 |
| Horizontal .....                         | 1743 |
| Vertical.....                            | 1743 |
| Appearance and Styling .....             | 1744 |
| Adjusting rotator item size.....         | 1744 |
| Image with Contents.....                 | 1745 |
| Display items.....                       | 1748 |
| Animation .....                          | 1751 |
| Theme .....                              | 1752 |
| RTL Support.....                         | 1754 |
| Keyboard interaction .....               | 1754 |
| Schedule .....                           | 1757 |
| Overview.....                            | 1757 |
| External and Internal Dependencies ..... | 1758 |

|  |      |
|--|------|
| Getting Started .....                      | 1760 |
| Create an HTML File .....                  | 1760 |
| Script/CSS References .....                | 1760 |
| Define Container to render Scheduler ..... | 1761 |
| Control Initialization.....                | 1761 |
| Setting TimeZone .....                     | 1762 |
| Data Binding.....                          | 1763 |
| Appointment Fields.....                    | 1763 |
| Appointment Field Validation .....         | 1765 |
| Binding to JSON Data Array .....           | 1767 |
| Binding Remote Data Service .....          | 1767 |
| OData V4.....                              | 1768 |
| WebAPI Binding .....                       | 1768 |
| Data binding using OLEDB.....              | 1769 |
| ASP.NET Web Method Binding.....            | 1774 |
| MVC Controller Action Binding .....        | 1782 |
| Loading Data on Demand.....                | 1785 |
| Views .....                                | 1788 |
| Day.....                                   | 1789 |
| Week.....                                  | 1790 |
| Work Week .....                            | 1790 |
| Month .....                                | 1791 |
| Custom.....                                | 1792 |
| Agenda .....                               | 1793 |
| Restriction on View Navigation.....        | 1794 |
| Timeline View .....                        | 1795 |
| Working with Appointments.....             | 1795 |
| Appointment Types.....                     | 1796 |
| CRUD operation .....                       | 1796 |
| Handling Appointment Actions .....         | 1805 |
| Read Only.....                             | 1806 |
| Drag and Drop.....                         | 1807 |
| Resize .....                               | 1812 |
| Categorization.....                        | 1814 |
| Priority .....                             | 1816 |

|  |      |
|--|------|
| Search or Filter Appointments .....                  | 1818 |
| Recurrence Options .....                             | 1821 |
| Reminder .....                                       | 1825 |
| Block Time Intervals .....                           | 1826 |
| Context Menu .....                                   | 1831 |
| Default Menu Options .....                           | 1831 |
| Custom Menu Options .....                            | 1833 |
| Handling Menu Actions .....                          | 1834 |
| Adding Categorize Option .....                       | 1835 |
| Remote Data Binding for Categorize .....             | 1836 |
| Resources .....                                      | 1837 |
| Fields of Resources .....                            | 1838 |
| Data Binding .....                                   | 1840 |
| Multiple Resources (Without Grouping) .....          | 1842 |
| Grouping .....                                       | 1843 |
| Different Working days and Hours for Resources ..... | 1846 |
| Customization .....                                  | 1847 |
| Hour Customization .....                             | 1848 |
| TimeScale .....                                      | 1849 |
| Hide Weekend days .....                              | 1850 |
| Date Customization .....                             | 1851 |
| Appointment Window Customization .....               | 1852 |
| Scheduler Customization using queryCellInfo .....    | 1859 |
| Navigation .....                                     | 1861 |
| View Navigation .....                                | 1861 |
| Date Navigation .....                                | 1862 |
| Appointment Navigation .....                         | 1863 |
| Template .....                                       | 1863 |
| Appointment Template .....                           | 1864 |
| Cell Templates .....                                 | 1865 |
| Date Header Template .....                           | 1866 |
| Resource Header Template .....                       | 1867 |
| TimeScale Templates .....                            | 1870 |
| Priority Settings Template .....                     | 1871 |
| Tooltip Template .....                               | 1872 |

|   |      |
|---|------|
| Agenda View Templates.....                              | 1873 |
| Globalization and Localization .....                    | 1875 |
| Globalization .....                                     | 1875 |
| Localization .....                                      | 1875 |
| Time Zone .....   | 1878 |
| Time Mode.....  | 1881 |
| Date Format .....                                       | 1882 |
| First Day of Week .....                                 | 1882 |
| Keyboard Navigation.....                                | 1883 |
| Setting Dimension.....                                  | 1884 |
| Scheduler Dimension .....                               | 1884 |
| Scheduler Cell Dimensions .....                         | 1885 |
| Responsiveness.....                                     | 1887 |
| Auto-Resizing Scheduler .....                           | 1887 |
| Scheduler in Mobile/Tablets .....                       | 1888 |
| Persistence.....  | 1889 |
| Export and Print .....                                  | 1890 |
| Export Appointments to ICS file .....                   | 1890 |
| PDF Export .....  | 1893 |
| Excel Export .....                                      | 1897 |
| Print .....   | 1899 |
| Import Appointments .....                               | 1901 |
| Miscellaneous .....                                     | 1903 |
| Time Indicator .....                                    | 1903 |
| Show/Hide All-Day Row .....                             | 1904 |
| Show/Hide Header bar.....                               | 1904 |
| Show/Hide TimeScale .....                               | 1905 |
| Show/Hide Location Field .....                          | 1905 |
| Recurrence Editor .....                                 | 1906 |
| Getting Started .....                                   | 1906 |
| Control Initialization.....                             | 1907 |
| Generating Recurrence Rule .....                        | 1907 |
| How To.....   | 1908 |
| Validate the Custom Appointment Window Fields .....     | 1908 |
| Highlight Different Work Hours for Each Resources ..... | 1909 |

|  |      |
|--|------|
| Display Scheduler with Appointments Filtered by Subject..... | 1910 |
| Customize the Default Appointment Window .....               | 1913 |
| Synchronize the Schedule with Outlook .....                  | 1914 |
| Scroller.....  | 1917 |
| Overview.....  | 1917 |
| Key Features .....   | 1917 |
| Create a simple Scroller in TypeScript.....                  | 1917 |
| Scroller Styles.....   | 1919 |
| Button Size .....  | 1919 |
| Scroller Size.....   | 1919 |
| Scroll Top .....   | 1919 |
| Scroll Left .....  | 1919 |
| Height .....   | 1919 |
| Width .....  | 1919 |
| Thumb Scrolling .....  | 1922 |
| Customizing the scroll Step .....                            | 1923 |
| RTL .....  | 1925 |
| Signature.....   | 1928 |
| Overview.....  | 1928 |
| Key Features .....   | 1928 |
| Getting Started .....  | 1928 |
| Creating an Signature in Typescript .....                    | 1928 |
| Adjusting Signature Size.....                                | 1929 |
| Signature Customization .....                                | 1930 |
| Background color .....                                       | 1930 |
| Background Image .....                                       | 1931 |
| Stroke color.....  | 1932 |
| Stroke Width.....  | 1933 |
| How To.....  | 1934 |
| Slider .....   | 1937 |
| Overview.....  | 1937 |
| Getting Started .....  | 1938 |
| Creating an Slider in TypeScript .....                       | 1938 |
| set min and max values.....                                  | 1939 |
| Behavior Settings .....                                      | 1940 |

|                                      |      |
|--------------------------------------|------|
| Height .....                         | 1940 |
| Width .....                          | 1940 |
| IncrementStep .....                  | 1940 |
| ReadOnly.....                        | 1941 |
| Persistence Support .....            | 1942 |
| Orientation .....                    | 1942 |
| Slider Types.....                    | 1943 |
| Updating slider value .....          | 1945 |
| Value .....                          | 1945 |
| Values .....                         | 1945 |
| MinValue.....                        | 1946 |
| MaxValue .....                       | 1946 |
| Buttons .....                        | 1947 |
| Scale Settings .....                 | 1948 |
| Show Scale .....                     | 1948 |
| Enable Small Ticks .....             | 1949 |
| Small step.....                      | 1950 |
| Large step .....                     | 1950 |
| Appearance and Styling .....         | 1951 |
| CSS Class .....                      | 1953 |
| Show Tooltip .....                   | 1954 |
| Show Rounded Corner .....            | 1955 |
| Animation .....                      | 1956 |
| Enabling Animation.....              | 1956 |
| Customizing Animation speed.....     | 1956 |
| Enable/Disable the Slider .....      | 1957 |
| Enabled .....                        | 1957 |
| RTL support.....                     | 1958 |
| Enabling RTL.....                    | 1958 |
| Keyboard Interaction .....           | 1959 |
| Configure keyboard interaction ..... | 1959 |
| Sparkline .....                      | 1960 |
| Overview.....                        | 1960 |
| Getting Started .....                | 1960 |
| Create your sparkline.....           | 1960 |

|                                    |      |
|------------------------------------|------|
| Populate Sparkline with data ..... | 1961 |
| Sparkline Type.....                | 1962 |
| Enable Tooltip .....               | 1962 |
| Working with Data .....            | 1963 |
| Local Data .....                   | 1963 |
| Sparkline Dimensions.....          | 1964 |
| Set size for the container .....   | 1964 |
| Set size in pixels .....           | 1964 |
| Responsive Sparkline .....         | 1964 |
| Sparkline Types .....              | 1965 |
| Line Type .....                    | 1965 |
| Column Type .....                  | 1965 |
| Area Type .....                    | 1966 |
| WinLoss Type .....                 | 1966 |
| Pie Type .....                     | 1966 |
| Axis Customize .....               | 1967 |
| Marker Customization .....         | 1967 |
| Point Customization .....          | 1968 |
| Tooltip.....                       | 1968 |
| Tooltip Customization .....        | 1969 |
| Tooltip Template.....              | 1969 |
| Range Band .....                   | 1970 |
| Sparkline Customization .....      | 1970 |
| Sparkline background .....         | 1971 |
| Stroke color and width.....        | 1971 |
| Sparkline border .....             | 1971 |
| Opacity.....                       | 1972 |
| Localization .....                 | 1972 |
| Padding for Sparkline .....        | 1972 |
| Canvas support .....               | 1972 |
| Themes .....                       | 1973 |
| Methods.....                       | 1973 |
| Events .....                       | 1973 |
| SplitButton .....                  | 1976 |
| Overview .....                     | 1976 |

|   |      |
|---|------|
| Create a simple SplitButton in TypeScript .....       | 1976 |
| Configuring SplitButton Properties .....              | 1977 |
| Easy customization .....                              | 1978 |
| Content for Split button .....                        | 1978 |
| Button Size .....                                     | 1979 |
| Content Type .....                                    | 1982 |
| Image Position .....                                  | 1985 |
| Theme support.....                                    | 1989 |
| Custom CSS .....                                      | 1990 |
| Dropdown Button .....                                 | 1993 |
| Arrow Position .....                                  | 1994 |
| RTL Support.....                                      | 1996 |
| Miscellaneous .....                                   | 1997 |
| Text .....  | 1997 |
| Show Rounded Corner .....                             | 1998 |
| Splitter .....  | 1999 |
| Overview .....  | 1999 |
| Key Features .....                                    | 1999 |
| Getting Started .....                                 | 1999 |
| Create a Splitter .....                               | 2000 |
| Configure Splitter Panes.....                         | 2001 |
| Configure Tree View.....                              | 2002 |
| Set Actions .....                                     | 2002 |
| Configure Collapsible and Expandable properties ..... | 2003 |
| Enabling Collapsible .....                            | 2003 |
| Disable Expandable .....                              | 2005 |
| Splitter Orientation .....                            | 2006 |
| Configure Splitter Orientation.....                   | 2006 |
| Nested Splitter Support .....                         | 2007 |
| Configure Nested Splitter.....                        | 2007 |
| Splitter Integration.....                             | 2009 |
| Configuring other widgets in Splitter .....           | 2009 |
| Appearance and Styling .....                          | 2011 |
| Responsive .....                                      | 2011 |
| Enabling Auto Resize.....                             | 2011 |



|  |      |
|--|------|
| Animation Support.....                 | 2012 |
| Adjusting Splitter Size .....          | 2013 |
| Resizable .....                        | 2014 |
| Theme .....                            | 2016 |
| Keyboard Navigation.....               | 2018 |
| Configuring Keyboard Navigation .....  | 2019 |
| RTL Support.....                       | 2020 |
| Enable RTL.....                        | 2020 |
| Spreadsheet.....                       | 2021 |
| Overview .....                         | 2021 |
| Getting started .....                  | 2022 |
| Adding Script Reference.....           | 2022 |
| Initialize Spreadsheet.....            | 2023 |
| Populate Spreadsheet with data .....   | 2024 |
| Apply Conditional Formatting .....     | 2025 |
| Export Spreadsheet as Excel File ..... | 2026 |
| SunburstChart.....                     | 2026 |
| Overview .....                         | 2026 |
| Getting Started .....                  | 2027 |
| Create Sunburst Chart.....             | 2027 |
| Populate Data source:.....             | 2028 |
| Initialize Sunburst Chart.....         | 2029 |
| Add Title to the Sunburst Chart .....  | 2030 |
| Enable Legend.....                     | 2030 |
| Add Data Labels .....                  | 2031 |
| Sunburst Elements .....                | 2032 |
| Start and End Angle.....               | 2032 |
| Sunburst Radius .....                  | 2032 |
| Sunburst Inner Radius .....            | 2033 |
| Levels .....                           | 2034 |
| GroupMemberPath .....                  | 2034 |
| Legend .....                           | 2035 |
| Legend Icon.....                       | 2036 |
| Positioning the Legend.....            | 2037 |
| Legend Item Size and border .....      | 2038 |

|  |      |
|--|------|
| Legend Size .....                        | 2039 |
| Legend Row and Columns .....             | 2040 |
| LegendInteractivity .....                | 2041 |
| ToggleSegmentSelection.....              | 2041 |
| Toggle Segment Visibility .....          | 2042 |
| Data Labels.....                         | 2043 |
| Label Overflow mode .....                | 2044 |
| Label Rotation Mode.....                 | 2046 |
| Customizing the data labels .....        | 2048 |
| Tooltip.....                             | 2049 |
| Tooltip Template.....                    | 2050 |
| Highlight.....                           | 2051 |
| Highlight Display mode .....             | 2052 |
| Highlight Mode .....                     | 2053 |
| Selection .....                          | 2057 |
| Selection Display mode .....             | 2058 |
| Selection Mode .....                     | 2059 |
| Zooming.....                             | 2063 |
| Zooming toolbar .....                    | 2064 |
| Animation .....                          | 2065 |
| Animation Types .....                    | 2066 |
| Appearance.....                          | 2067 |
| Palette.....                             | 2067 |
| Built- in Themes .....                   | 2068 |
| Methods.....                             | 2069 |
| Events .....                             | 2070 |
| Tab.....                                 | 2074 |
| Overview.....                            | 2074 |
| Getting Started .....                    | 2075 |
| Create Tab Control.....                  | 2075 |
| Configure Content.....                   | 2076 |
| Create the Rating .....                  | 2076 |
| AJAX Content Load (Load On Demand) ..... | 2077 |
| Orientation Change.....                  | 2079 |
| Header Image Customization .....         | 2080 |

|   |      |
|---|------|
| Configuring Contents to remaining Tab items ..... | 2080 |
| Behavior Settings .....                           | 2083 |
| Close Button.....                                 | 2083 |
| Orientation.....                                  | 2084 |
| State Maintenance.....                            | 2085 |
| Appearance and Styling .....                      | 2086 |
| Header Image Customization .....                  | 2086 |
| Rounded corner .....                              | 2087 |
| Enable/Disable .....                              | 2088 |
| Enabling Reload Icon.....                         | 2089 |
| Collapsible Tabs .....                            | 2090 |
| Adjusting Tab Size .....                          | 2091 |
| Theme .....                                       | 2093 |
| Custom styles.....                                | 2094 |
| Integration with other widgets .....              | 2095 |
| AJAX Content Load (Load on Demand) .....          | 2097 |
| Sub Tab with AJAX Content.....                    | 2097 |
| RTL Support.....                                  | 2100 |
| Keyboard Navigation.....                          | 2101 |
| Scroll Support.....                               | 2102 |
| Template Support .....                            | 2104 |
| TagCloud .....                                    | 2105 |
| Overview .....                                    | 2105 |
| Getting Started .....                             | 2105 |
| Create TagCloud widget .....                      | 2106 |
| Set Min and Max Font Size.....                    | 2107 |
| Set event to perform an operation .....           | 2108 |
| Data-Binding .....                                | 2109 |
| Fields.....                                       | 2109 |
| Local Binding.....                                | 2110 |
| Remote Binding.....                               | 2111 |
| Title Customization .....                         | 2112 |
| Show title .....                                  | 2112 |
| Title text.....                                   | 2113 |
| Title image .....                                 | 2114 |

|  |      |
|--|------|
| Appearance and Styling .....             | 2115 |
| Minimum and maximum Font size .....      | 2115 |
| Tag format .....                         | 2116 |
| Theme .....                              | 2118 |
| CssClass .....                           | 2119 |
| RTL Support .....                        | 2120 |
| Enabling RTL Support .....               | 2120 |
| Tile .....                               | 2121 |
| Overview .....                           | 2121 |
| Key Features .....                       | 2121 |
| Getting Started .....                    | 2121 |
| Create a Tile .....                      | 2121 |
| Image Configuration .....                | 2125 |
| Text Configuration .....                 | 2127 |
| Template Support .....                   | 2128 |
| Configure Badge .....                    | 2129 |
| LiveTile Configuration .....             | 2130 |
| Customize size .....                     | 2132 |
| Add Group Tiles .....                    | 2133 |
| TimePicker .....                         | 2136 |
| Overview .....                           | 2136 |
| Key Features .....                       | 2136 |
| Getting Started .....                    | 2136 |
| Create your first TimePicker .....       | 2136 |
| DisableTimeRanges .....                  | 2137 |
| Behavior Settings .....                  | 2138 |
| Set value of the TimePicker widget ..... | 2138 |
| Enable/Disable TimePicker widget .....   | 2138 |
| Restrict editing .....                   | 2139 |
| Rounded Corner .....                     | 2140 |
| Scaling .....                            | 2140 |
| State persistence .....                  | 2141 |
| Strict mode of the TimePicker .....      | 2142 |
| Interval .....                           | 2142 |
| Range .....                              | 2143 |

|   |      |
|---|------|
| Steps to change minTime & maxTime of the TimePicker ..... | 2143 |
| TimePicker Customization .....                            | 2144 |
| Creating TimePicker Widget.....                           | 2144 |
| Time Format.....  | 2145 |
| Steps to change Time Format of TimePicker widget .....    | 2145 |
| Globalization.....  | 2145 |
| Enabling Globalization Support.....                       | 2145 |
| RTL .....   | 2147 |
| Enabling Right-To-Left Support .....                      | 2147 |
| Keyboard Interaction .....                                | 2148 |
| Configure Keyboard Interaction .....                      | 2148 |
| ToggleButton .....  | 2149 |
| Overview .....  | 2149 |
| Getting Started .....                                     | 2149 |
| Creating an ToggleButton in TypeScript .....              | 2149 |
| Easy Customization .....                                  | 2150 |
| Toggle State .....  | 2150 |
| Toggle state with icons .....                             | 2151 |
| Toggle button size.....                                   | 2154 |
| Content type .....  | 2156 |
| Image position .....                                      | 2160 |
| Theme support.....  | 2165 |
| Custom CSS .....  | 2165 |
| Button types .....  | 2168 |
| RTL support.....  | 2169 |
| Miscellaneous .....                                       | 2170 |
| Show Rounded Corner .....                                 | 2170 |
| Prevent Toggle .....                                      | 2171 |
| Toolbar.....  | 2172 |
| Overview .....  | 2172 |
| Getting Started .....                                     | 2172 |
| Create Toolbar for PDF Reader .....                       | 2172 |
| Create a Toolbar .....                                    | 2172 |
| Initialize Toolbar Items.....                             | 2173 |
| Render remaining Toolbar items.....                       | 2175 |

|   |      |
|---|------|
| Add Actions to Toolbar Items.....               | 2177 |
| Data binding.....                               | 2179 |
| Data fields and configuration .....             | 2179 |
| Local data .....                                | 2180 |
| Remote data .....                               | 2182 |
| Behavior settings .....                         | 2182 |
| Enabling Toolbar .....                          | 2182 |
| Hiding Toolbar.....                             | 2184 |
| Orientation .....                               | 2184 |
| Horizontal .....                                | 2185 |
| Vertical.....                                   | 2185 |
| Appearance and Styling .....                    | 2187 |
| Adjusting Toolbar size .....                    | 2187 |
| Enabling Rounded Corner .....                   | 2187 |
| Enabling Separator .....                        | 2188 |
| Themes .....                                    | 2188 |
| CssClass.....                                   | 2188 |
| Template Support .....                          | 2189 |
| Through Items API:.....                         | 2190 |
| Through template field in dataSource API: ..... | 2191 |
| RTL .....                                       | 2192 |
| Keyboard Navigation.....                        | 2192 |
| Responsive Layout .....                         | 2193 |
| responsiveType:Inline .....                     | 2196 |
| responsiveType:Popup.....                       | 2197 |
| Tooltip.....                                    | 2197 |
| Overview.....                                   | 2197 |
| Getting started.....                            | 2198 |
| Preparing HTML document .....                   | 2198 |
| Create a Tooltip .....                          | 2198 |
| Setting Dimensions .....                        | 2200 |
| Tooltip Appearance.....                         | 2200 |
| Customization .....                             | 2201 |
| Template Support .....                          | 2201 |
| Animation Effects.....                          | 2204 |

|   |      |
|---|------|
| Modernize the tooltip's content .....           | 2205 |
| Closing Mode .....                              | 2206 |
| Position .....                                  | 2207 |
| Containment .....                               | 2209 |
| Associates .....                                | 2209 |
| Collision.....                                  | 2211 |
| How To.....                                     | 2212 |
| Use AJAX to generate the Tooltip's content..... | 2212 |
| Integration with the Slider control .....       | 2215 |
| Tip(arrow) customization .....                  | 2216 |
| Initialize Tooltip for the target element ..... | 2218 |
| Interact with the Tooltip .....                 | 2220 |
| TreeGrid.....                                   | 2220 |
| Overview.....                                   | 2220 |
| Getting Started .....                           | 2221 |
| Create your first TreeGrid in TypeScript.....   | 2221 |
| TreeMap .....                                   | 2227 |
| Overview.....                                   | 2227 |
| Key Features .....                              | 2228 |
| Use Case Scenarios .....                        | 2228 |
| Getting Started .....                           | 2228 |
| Create a TreeMap .....                          | 2229 |
| Add Libraries .....                             | 2229 |
| Initialize TreeMap .....                        | 2230 |
| GroupTreeMap Items using Levels.....            | 2231 |
| Customize TreeMap Appearance by Range .....     | 2232 |
| Enable Tooltip .....                            | 2233 |
| Legend .....                                    | 2234 |
| DataBinding .....                               | 2236 |
| TreeMapLevels.....                              | 2237 |
| Flat Level .....                                | 2237 |
| Hierarchical Level .....                        | 2238 |
| Layout .....                                    | 2239 |
| Squarified.....                                 | 2240 |
| SliceAndDiceAuto.....                           | 2240 |

|  |      |
|--|------|
| SliceAndDiceHorizontal .....             | 2241 |
| SliceAndDiceVertical .....               | 2242 |
| Customization .....                      | 2243 |
| Color .....                              | 2243 |
| Tooltip.....                             | 2246 |
| Leaf Item Setting.....                   | 2246 |
| Border Brush .....                       | 2248 |
| Border Thickness.....                    | 2248 |
| Dock Position .....                      | 2248 |
| Clicking and Dragging.....               | 2249 |
| Fill with Gradient.....                  | 2249 |
| Responsive Treemap .....                 | 2249 |
| GroupColorMapping .....                  | 2250 |
| GroupSelectionMode.....                  | 2250 |
| Header .....                             | 2250 |
| Specifying HierarchicalDatasource .....  | 2250 |
| Localization .....                       | 2250 |
| Treemap Items.....                       | 2251 |
| TreeMap Elements.....                    | 2251 |
| Legend .....                             | 2251 |
| Header .....                             | 2253 |
| Customizing the header .....             | 2254 |
| Label.....                               | 2255 |
| Customizing the Overflow labels.....     | 2256 |
| Palette Color Mapping .....              | 2257 |
| Drill Down Support.....                  | 2257 |
| Enable Drill Down.....                   | 2257 |
| AngularJS Support.....                   | 2259 |
| Methods.....                             | 2261 |
| Events .....                             | 2262 |
| TreeView .....                           | 2263 |
| Overview.....                            | 2263 |
| Key features .....                       | 2263 |
| Getting Started .....                    | 2264 |
| Creating an TreeView in TypeScript ..... | 2264 |



|   |      |
|---|------|
| Drag and Drop.....                                  | 2266 |
| Checkboxes .....                                    | 2267 |
| Populate Data .....                                 | 2267 |
| Fields.....   | 2267 |
| Local Data .....                                    | 2269 |
| Remote Data .....                                   | 2271 |
| Load on Demand.....                                 | 2272 |
| Tree Node .....                                     | 2274 |
| Get/Set Node Value .....                            | 2274 |
| Get Parent Node .....                               | 2275 |
| Get Node Index .....                                | 2275 |
| Node Manipulations .....                            | 2275 |
| Editing.....  | 2277 |
| Selection .....                                     | 2278 |
| Ensure Visibility.....                              | 2278 |
| Checkboxes .....                                    | 2279 |
| Indeterminate Checkboxes .....                      | 2279 |
| Auto Checkable .....                                | 2280 |
| Check or Uncheck Node .....                         | 2280 |
| Get Checked Nodes.....                              | 2281 |
| Drag and drop .....                                 | 2282 |
| Position Indicators .....                           | 2282 |
| Restriction.....                                    | 2282 |
| Drag and Drop between Trees .....                   | 2283 |
| Auto Node Structuring .....                         | 2284 |
| Multiple Selection .....                            | 2285 |
| Select Nodes .....                                  | 2286 |
| Get Selected Nodes.....                             | 2286 |
| Drag and Drop Multiple Nodes .....                  | 2287 |
| Full Row Selection .....                            | 2288 |
| Template.....                                       | 2289 |
| State Persistence .....                             | 2290 |
| Keyboard Interaction .....                          | 2291 |
| How To.....   | 2292 |
| Update the modified data from tree to database..... | 2292 |

|   |      |
|---|------|
| TreeView context menu to process node operations .....                | 2292 |
| Sorted data using refresh method .....                                | 2292 |
| Persist updated data after edit, add and remove node .....            | 2292 |
| Filtering nodes in TreeView .....                                     | 2293 |
| AngularJS data binding to update data while add and remove node ..... | 2293 |
| Set Tooltip for TreeView nodes .....                                  | 2293 |
| Auto hide/show the expand/collapse icon of TreeView .....             | 2293 |
| Customize the expand/collapse icons of TreeView .....                 | 2293 |
| Uploadbox .....   | 2293 |
| Overview .....  | 2293 |
| Getting Started .....   | 2293 |
| Create Uploadbox in Typescript .....                                  | 2294 |
| Set Restriction for File Extension .....                              | 2297 |
| Upload Multiple Files .....   | 2299 |
| File Actions .....  | 2300 |
| Save File Action .....  | 2300 |
| Remove File Action .....  | 2302 |
| Auto Upload .....   | 2303 |
| Restricting uploading files based on its extension .....              | 2305 |
| Allow Extension .....   | 2305 |
| Deny Extension .....  | 2305 |
| File Size .....   | 2306 |
| Maximum File Size for Uploadbox .....                                 | 2306 |
| Maximum File Upload Size in IIS .....                                 | 2307 |
| Enables or Disables the Uploadbox .....                               | 2308 |
| Asynchronous Upload .....   | 2309 |
| Synchronous Upload .....  | 2309 |
| Multiple files upload .....   | 2311 |
| Drag and Drop Support .....   | 2313 |
| Enable drag and drop .....  | 2313 |
| Drag Area text .....  | 2314 |
| Adjust Drop area size .....   | 2315 |
| Drop area with Browse button behavior .....                           | 2316 |
| Appearance and styling .....  | 2317 |
| Customizing Button Text .....   | 2318 |

|  |      |
|--|------|
| Customizing Upload Dialog .....                  | 2319 |
| Show or Hide File details .....                  | 2320 |
| Theme .....                                      | 2321 |
| Custom CSS .....                                 | 2321 |
| RTL Support.....                                 | 2322 |
| WaitingPopup .....                               | 2323 |
| Overview .....                                   | 2323 |
| Getting Started .....                            | 2324 |
| Create Username and Password .....               | 2324 |
| Add WaitingPopup Widget.....                     | 2326 |
| Behavior and Settings .....                      | 2326 |
| Automatic Initializing WaitingPopup widget ..... | 2326 |
| Enable / Disable Popup Indicator .....           | 2328 |
| Show / Hide WaitingPopup .....                   | 2329 |
| Appearance and Styling .....                     | 2330 |
| Custom Text .....                                | 2330 |
| Template .....                                   | 2331 |
| CSS Class .....                                  | 2333 |

## NuGet Packages

NuGet is a package manager for the .NET framework. The NuGet client tools simplify the process of installing and upgrading packages. This can be used to automatically add files and references to your Visual Studio projects.

---

**Note:** You can use the Syncfusion TypeScript NuGet packages without installing the Essential Studio or TypeScript platform installation to implement the Syncfusion TypeScript controls.

---

### Get the Syncfusion NuGet feed URL

You should get the private Syncfusion JavaScript NuGet feed URL to install or upgrade the Syncfusion TypeScript NuGet packages. To get the URL from Syncfusion website use the following steps:

1. Navigate to [nuget.syncfusion.com](http://nuget.syncfusion.com), and select **WEB** tab.
2. Navigate to **WEB(Essential JS1)**, click the Copy URL label under JavaScript platform to copy the Syncfusion JavaScript platform NuGet feed to clipboard or directly use the following URL:

[http://nuget.syncfusion.com/nuget\\_javascript/nuget/getsyncfusionpackages/javascript](http://nuget.syncfusion.com/nuget_javascript/nuget/getsyncfusionpackages/javascript)

### JavaScript

Your JavaScript NuGet package URL

`http://nuget.syncfusion.com/nuget_javascript/nuget/getsyncfusionpackages/javascript`

[Copy URL](#)

3. Now, use this NuGet feed URL to access the Syncfusion NuGet Packages in Visual Studio.

### Add the Syncfusion NuGet feed URL

#### Windows

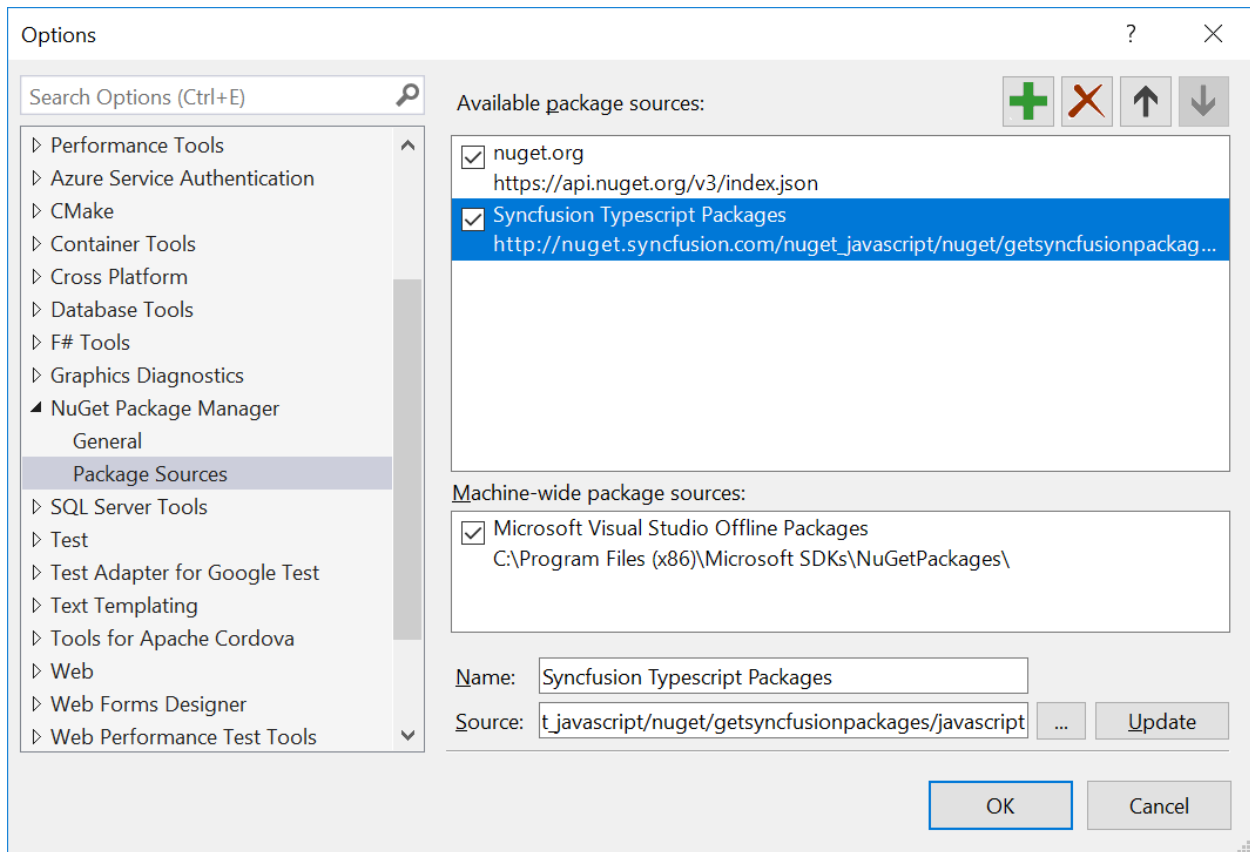
1. Open your Visual Studio application. 2. On the **Tools** menu, select **Options**. 3. Expand the **NuGet Package Manager** and select **Package Sources**. 4. Click the **Add** button (green plus), and enter the 'Package Name' and 'Package Source URL' of the Syncfusion TypeScript NuGet packages.

**Name:** Name of the package listed in the available package sources.

**Source:** Syncfusion JavaScript NuGet Feed URL

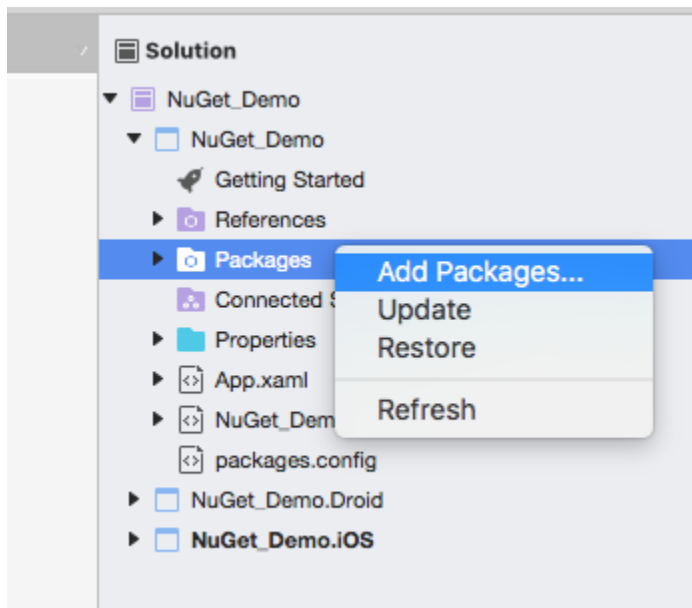
[http://nuget.syncfusion.com/nuget\\_javascript/nuget/getsyncfusionpackages/javascript](http://nuget.syncfusion.com/nuget_javascript/nuget/getsyncfusionpackages/javascript).

5. Click the **Update** button to add the name and source details to package sources.

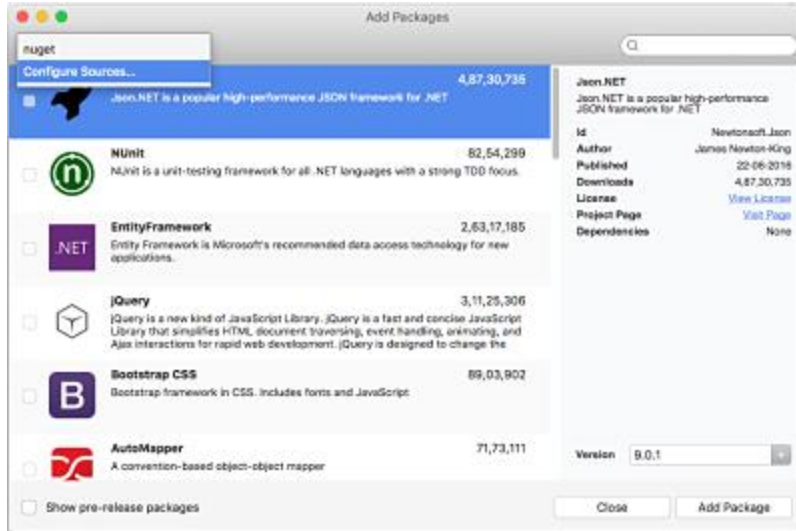


### macOS

1. Open your Visual Studio application. 2. Right-click on the Packages folder in the project, and then select **Add Packages...**



3. Choose the **Configure Sources...** from the dropdown that appears in the left corner of the Add Packages dialog.

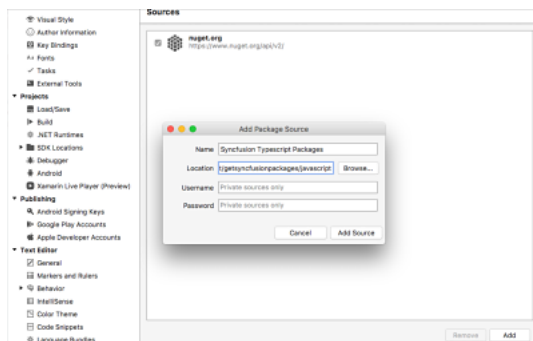


4. At the bottom right corner of the dialog, click the **Add** button to enter the feed name and the URL.

**Name:** Enter the name (For e.g., Syncfusion TypeScript Packages).

**Location:** Enter the following URL –

[http://nuget.syncfusion.com/nuget\\_javascript/nuget/getsyncfusionpackages/javascript](http://nuget.syncfusion.com/nuget_javascript/nuget/getsyncfusionpackages/javascript).



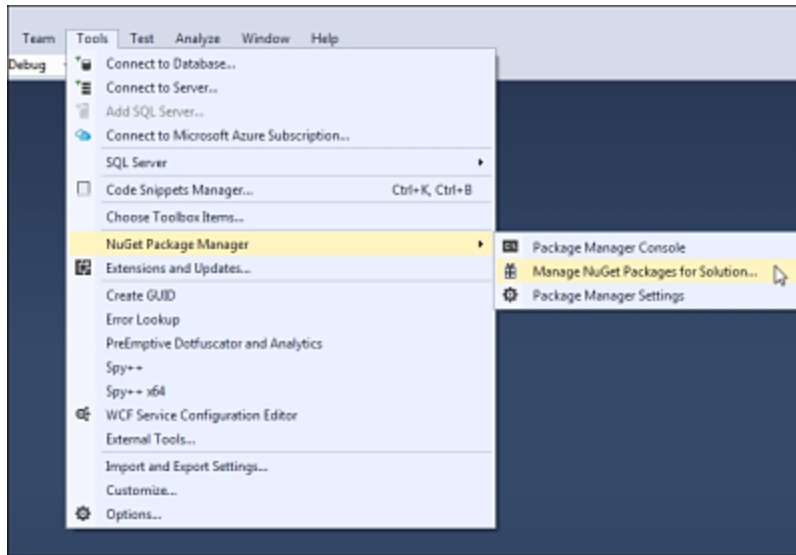
5. Now, click **Add Source** and then click **OK**.

## Installing NuGet Packages

### Using NuGet Package Manager

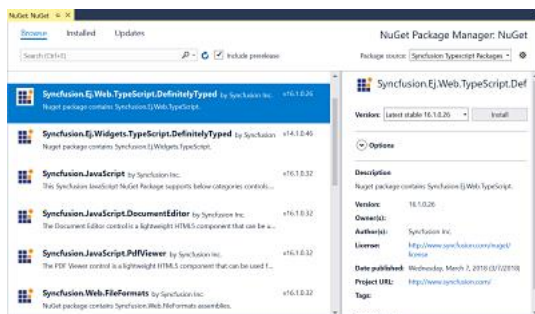
The NuGet Package Manager can be used to search and install NuGet packages in the Visual Studio solution or project:

1. On the **Tools**, menu, NuGet Package Manager | Manage NuGet Packages for Solution...



Alternatively, right-click on the project/solution in Solution Explorer tab, and choose **Manage NuGet Packages...**

2. By default, the NuGet.org package is selected in the **Package source** drop-down. Select your appropriate feed name that you configured.



3. The NuGet Packages are listed and available in the package source feed URL. Search and install the required packages in your application, by clicking **Install** button.

### Using Package Manager Console

To reference the Synfusion TypeScript component using the Package Manager Console as NuGet packages,

1. On the **Tools** menu, select **NuGet Package Manager** and then **Package Manager Console**. 2. Run the following NuGet installation commands:

### Accordion

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
```

```
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

## HTML

```
<div id="basicAccordion" style="width: 500px">
<h3>
<a href="#">Essential Studio ASP.NET</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
<ul>
<li>
<h4>DocIO</h4>
</li>
<li>
<h4>Pdf </h4>
</li>
<li>
<h4>Gauge </h4>
</li>
<li>
<h4>Schedule </h4>
</li>
<li>
<h4>Diagram </h4>
</li>
<li>
<h4>Tools </h4>
</li>
</ul>
</div>
<h3>
<a href="#">Essential Studio ASP.NET MVC</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid </h4>
</li>
<li>
<h4>Gantt </h4>
</li>
<li>
<h4>Schedule </h4>
</li>
<li>
```



```

<h4>Diagram </h4>
</li>
</ul>
</div>
<h3>
<a href="#">Essential Studio Javascript</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid </h4>
</li>
<li>
<h4>Gantt </h4>
</li>
<li>
<h4>Schedule </h4>
</li>
<li>
<h4>Diagram </h4>
</li>
</ul>
</div>
</div>

```

Create the Accordion control as follows.

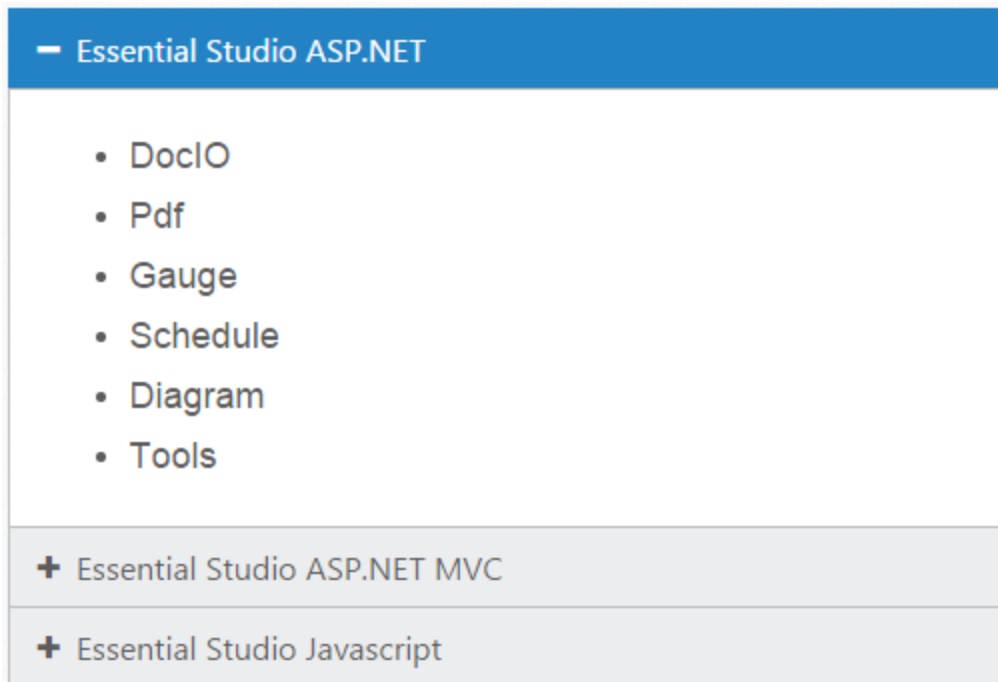
### HTML

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AccordionComponent {
$(function () {
var sample = new ej.Accordion($("#basicAccordion"));
});
}

```

You can execute the above code example to display the Accordion control with simple control list.



You can customize the Accordion control using various properties. The Accordion control properties and its default values are described in the following section.

#### Configure Multiple Open

You can have multiple **Accordion** tabs opened to view all products at a time. To achieve this set the **enableMultipleOpen** property of the **Accordion** control to true.

---

**Note:** enableMultipleOpen property is false by default.

---

You can also open all the panels during initialization using the **selectedItems** property of the **Accordion** control. The following code sample illustrates the opening of multiple tabs by passing the tab index values of tab.

#### HTML

```
<div id="basicAccordion" style="width: 500px">
  <h3>
    <a href="#">Essential Studio ASP.NET</a>
  </h3>
  <div>
    <!-- add accordion contents here to load contents under this header -->
    <ul>
      <li>
        <h4>DocIO</h4>
      </li>
      <li>
        <h4>Pdf </h4>
      </li>
      <li>
        <h4>Gauge </h4>
      </li>
      <li>
        <h4>Schedule </h4>
      </li>
    </ul>
  </div>
</div>
```

```

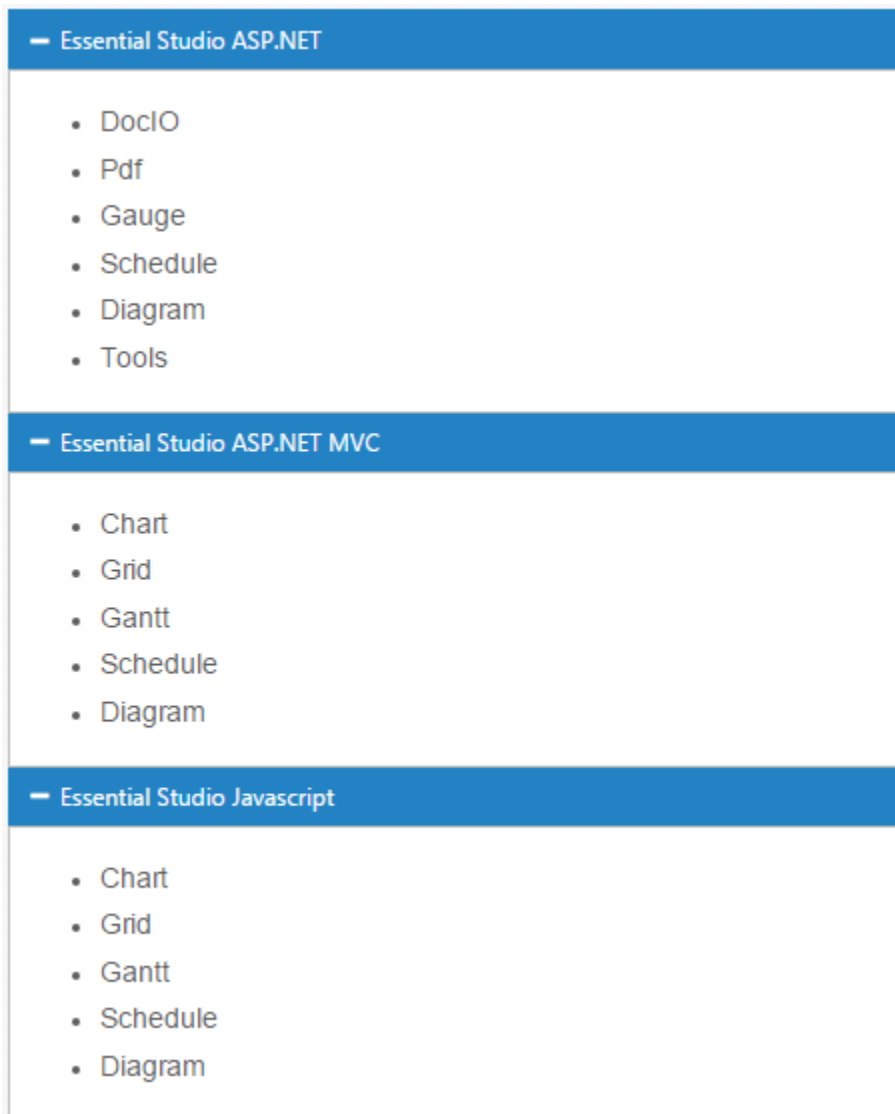
<li>
<h4>Diagram </h4>
</li>
<li>
<h4>Tools </h4>
</li>
</ul>
</div>
<h3>
<a href="#">Essential Studio ASP.NET MVC</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid </h4>
</li>
<li>
<h4>Gantt </h4>
</li>
<li>
<h4>Schedule </h4>
</li>
<li>
<h4>Diagram </h4>
</li>
</ul>
</div>
<h3>
<a href="#">Essential Studio Javascript</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid </h4>
</li>
<li>
<h4>Gantt </h4>
</li>
<li>
<h4>Schedule </h4>
</li>
<li>
<h4>Diagram </h4>
</li>
</ul>
</div>
</div>

```

## HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AccordionComponent {
$(function () {
var sample = new ej.Accordion($("#basicAccordion"), { enableMultipleOpen:
true, selectedItems:[0,1,2] });
});
}
```

**Accordion** control with **enableMultipleOpen** property is illustrated in the following screen shot.



### *Setting rounded corner*

**Accordion** control, by default, is rendered in a regular rectangle. You can modify the regular rectangles with rounded corners by setting the **showRoundedCorner** property to **True**.

**Note:** showRoundedCorner property is False by default.

## HTML

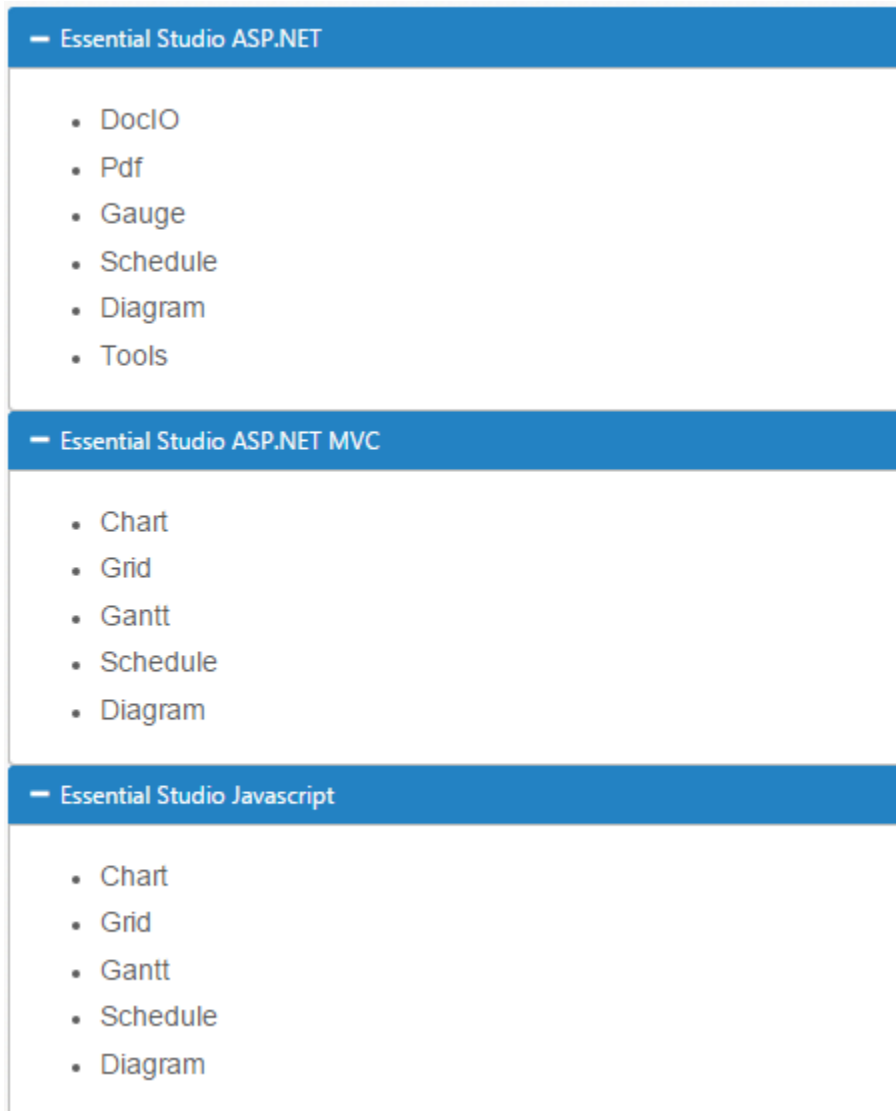
```
<div id="basicAccordion" style="width: 500px">
<h3>
<a href="#">Essential Studio ASP.NET</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
<ul>
<li>
<h4>DocIO</h4>
</li>
<li>
<h4>Pdf </h4>
</li>
<li>
<h4>Gauge </h4>
</li>
<li>
<h4>Schedule </h4>
</li>
<li>
<h4>Diagram </h4>
</li>
<li>
<h4>Tools </h4>
</li>
</ul>
</div>
<h3>
<a href="#">Essential Studio ASP.NET MVC</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid </h4>
</li>
<li>
<h4>Gantt </h4>
</li>
<li>
<h4>Schedule </h4>
</li>
<li>
<h4>Diagram </h4>
</li>
</ul>
</div>
<h3>
<a href="#">Essential Studio Javascript</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
```

```
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid </h4>
</li>
<li>
<h4>Gantt </h4>
</li>
<li>
<h4>Schedule </h4>
</li>
<li>
<h4>Diagram </h4>
</li>
</ul>
</div>
</div>
```

## HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AccordionComponent {
$(function () {
var sample = new ej.Accordion($("#basicAccordion"), { enableMultipleOpen:
true,showRoundedCorner: true, selectedItems:[0,1,2] });
});
}
```

The following screenshot illustrates the **Accordion** control with rounded corners.



### Customize Icon

You can customize the **Header** icon using **customIcon** property. This property has two features such as **header** and **selectedHeader**. By default, the classes of **header** and **selectedHeader** are **e-collapse** and **e-expand** respectively.

You can change the + and - symbols in the **Accordion** header, that are the default icons with Up or Down arrow icons.

Up or Down arrow icons are available in **e-arrowheadup** and **e-arrowheaddown** classes respectively in the ej.widgets.core.min.css stylesheets from the sample.

You can set the Up or Down arrow icon to **Accordion** header, by adding **e-arrowheadup** and **e-arrowheaddown** class to **selectedHeader** and **header** properties respectively.

### JAVASCRIPT

```
<div id="basicAccordion" style="width: 500px">
<h3>
<a href="#">Essential Studio ASP.NET</a>
```

```

</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
<ul>
<li>
<h4>DocIO</h4>
</li>
<li>
<h4>Pdf </h4>
</li>
<li>
<h4>Gauge </h4>
</li>
<li>
<h4>Schedule </h4>
</li>
<li>
<h4>Diagram </h4>
</li>
<li>
<h4>Tools </h4>
</li>
</ul>
</div>
<h3>
<a href="#">Essential Studio ASP.NET MVC</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid </h4>
</li>
<li>
<h4>Gantt </h4>
</li>
<li>
<h4>Schedule </h4>
</li>
<li>
<h4>Diagram </h4>
</li>
</ul>
</div>
<h3>
<a href="#">Essential Studio Javascript</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
<ul>
<li>
<h4>Chart </h4>
</li>
<li>

```

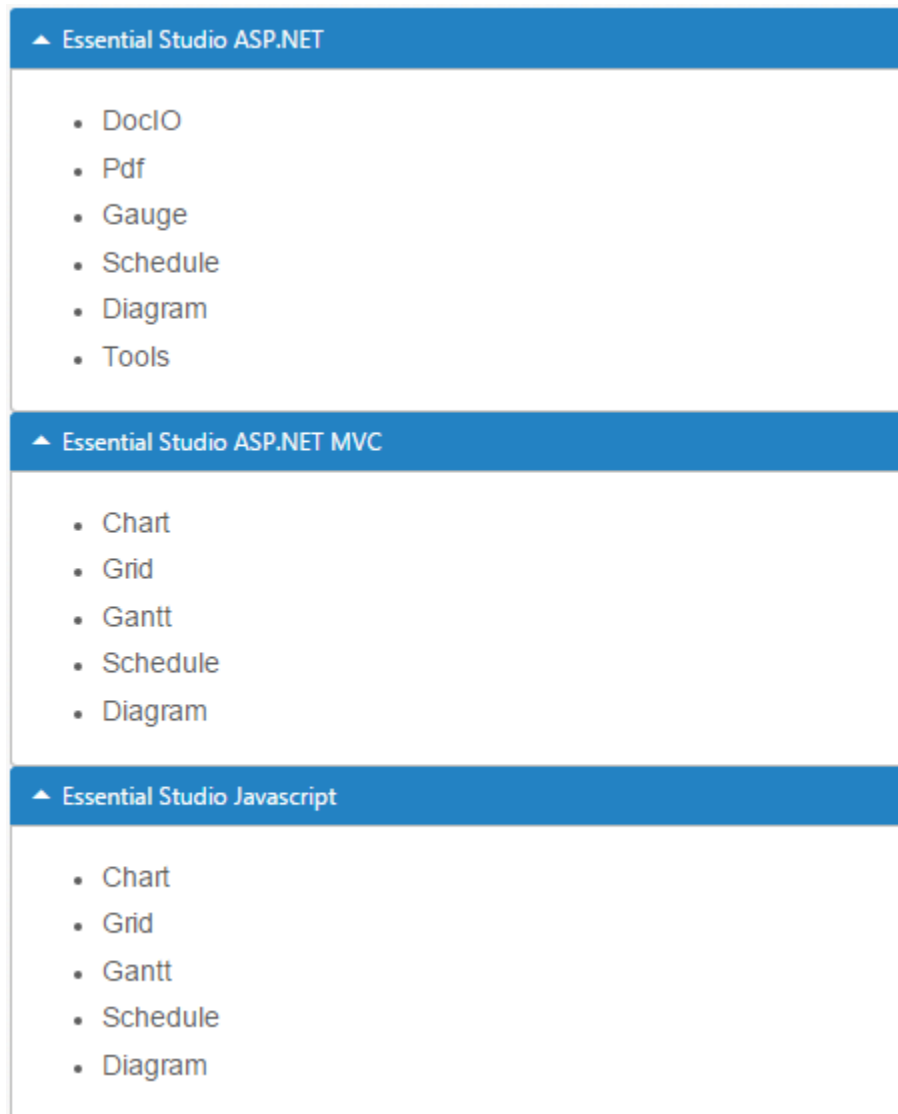


```
<h4>Grid </h4>
</li>
<li>
<h4>Gantt </h4>
</li>
<li>
<h4>Schedule </h4>
</li>
<li>
<h4>Diagram </h4>
</li>
</ul>
</div>
</div>
```

## HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AccordionComponent {
$(function () {
var sample = new ej.Accordion($("#basicAccordion"), {
enableMultipleOpen: true, showRoundedCorner: true, selectedItems: [0, 1, 2],
customIcon: {
header: "e-arrowheaddown",
selectedHeader: "e-arrowheadup"
}
});
});
}
```

The following screenshot illustrates the customization of **selectedHeader** and **header** of the **Accordion** control.



## Header customization

### Collapsible

**Accordion** widget allows you to set Collapsible state for an **Accordion** header. Thus you can expand and collapse accordion contents. By default **collapsible** is set to **false**.

### Enable Collapsible settings

The following steps explains to enable Collapsible state for **Accordion**.

In an HTML page, define a div element that is a container for Accordion widget and add the contents correspondingly

### HTML

```
<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
```

```

Orubase is the only mobile application development Framework built
especially for developing complex line-of-business mobile applications
targeting iOS, Android, and Windows Phone platforms in the shortest possible
time frame.
</div>
<h3>
<a href="#">WinRTXAML</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Essential Studio for WinRT contains all the controls you need to build line-
of-business tablet applications including grid, chart, map, tree map, SSRS
report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and
much more. It also includes a unique set of controls for reading and writing
Excel, Word, and PDF documents in Windows store apps.
</div>
<h3>
<a href="#">Metro Studio</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Syncfusion Metro Studio is a collection of over 2500 Metro-style icon
templates that can be easily customized to create thousands of unique Metro
icons.
</div>
</div>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AccordionComponent {
$(function () {
// Configure collapsible header for Accordion
var sample = new ej.Accordion($("#accordion"), {
collapsible: true
});
});
}

```

Output for Accordion control with collapsible headers.



Enable Header expand

**Accordion** widget provides you support to set the event, where the headers should expand and collapse. The **events** properties takes default events like mouseout, mouseover, and click.

### Configure header expand event

The following steps explain you to configure header expand event for **Accordion**.

In an HTML page, define a div element that is a container for Accordion widget and add the contents correspondingly

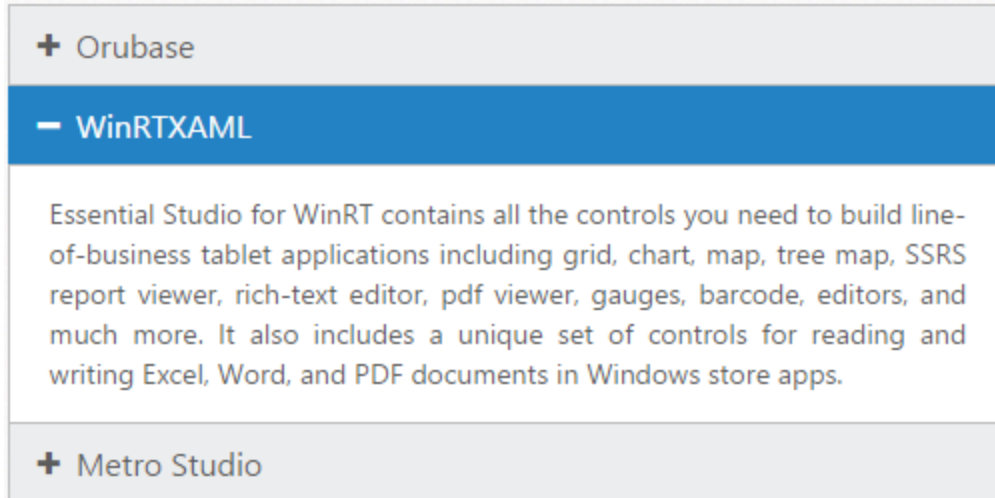
### HTML

```
<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Orubase is the only mobile application development Framework built
especially for developing complex line-of-business mobile applications
targeting iOS, Android, and Windows Phone platforms in the shortest possible
time frame.
</div>
<h3>
<a href="#">WinRTXAML</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Essential Studio for WinRT contains all the controls you need to build line-
of-business tablet applications including grid, chart, map, tree map, SSRS
report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and
much more. It also includes a unique set of controls for reading and writing
Excel, Word, and PDF documents in Windows store apps.
</div>
<h3>
<a href="#">Metro Studio</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Syncfusion Metro Studio is a collection of over 2500 Metro-style icon
templates that can be easily customized to create thousands of unique Metro
icons.
</div>
</div>
```

### JAVASCRIPT

```
// Configure header expand event for Accordion
var sample = new ej.Accordion($("#accordion"), {
  events: "mouseout"
});
```

Output for Accordion control that expands header on mouseout event is as follows.



Set selected header

#### Single selection

Using **selectedIndex** property you can modify the expanded panel when the control is rendered. By default **selectedIndex** is '0' that always activate the first **Accordion** panel.

#### Specify the selected item in Accordion panel

The following steps explains you to configure selected item for **Accordion**.

In an HTML page, define a div element that is a container for Accordion widget and add the contents correspondingly

#### HTML

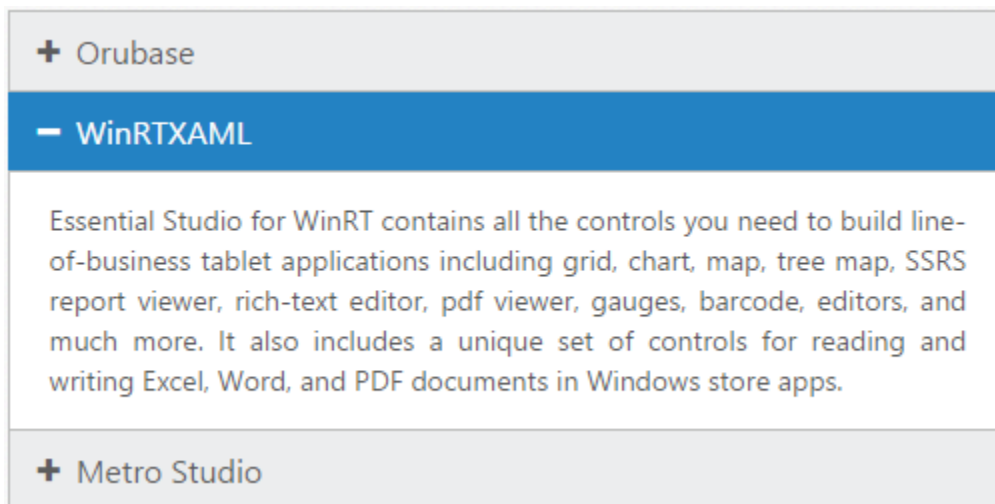
```
<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Orubase is the only mobile application development Framework built
especially for developing complex line-of-business mobile applications
targeting iOS, Android, and Windows Phone platforms in the shortest possible
time frame.
</div>
<h3>
<a href="#">WinRTXAML</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Essential Studio for WinRT contains all the controls you need to build line-
of-business tablet applications including grid, chart, map, tree map, SSRS
report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and
much more. It also includes a unique set of controls for reading and writing
Excel, Word, and PDF documents in Windows store apps.
</div>
<h3>
<a href="#">Metro Studio</a>
</h3>
<div>
```

```
<!-- add accordion contents here to load contents under this header -->
Syncfusion Metro Studio is a collection of over 2500 Metro-style icon
templates that can be easily customized to create thousands of unique Metro
icons.
</div>
</div>
```

## JAVASCRIPT

```
// Configure selected item for Accordion based on the index
var sample = new ej.Accordion($("#accordion"), {
  selectedItemIndex: 1
});
```

Output for Accordion control with the selected item by index is as follows.



## Multiple selection

In **Accordion** widget you can select multiple panel items using **selectedItems** property. It takes array of indices that needs to be selected on rendering the control. As you need to select multiple items, you can set **enableMultipleOpen** to **true**.

### Configure multiple selection in Accordion panel

The following steps explains to configure selected items for **Accordion**.

In an HTML page, define a <div> element that is a container for Accordion widget and add the contents correspondingly

## HTML

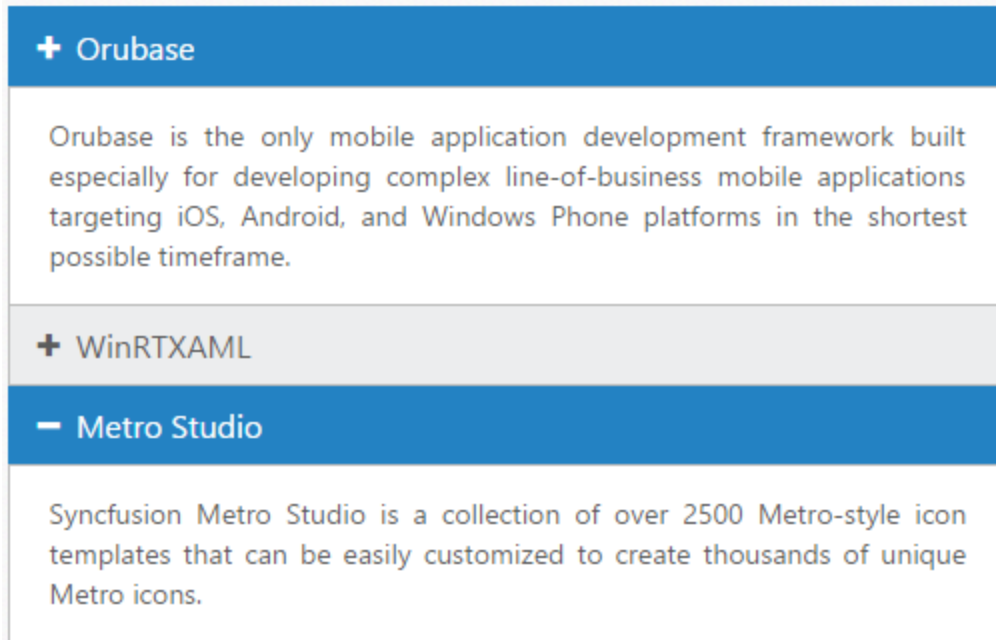
```
<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Orubase is the only mobile application development Framework built
especially for developing complex line-of-business mobile applications
```

```
targeting iOS, Android, and Windows Phone platforms in the shortest possible
time frame.
</div>
<h3>
<a href="#">WinRTXAML</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Essential Studio for WinRT contains all the controls you need to build line-
of-business tablet applications including grid, chart, map, tree map, SSRS
report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and
much more. It also includes a unique set of controls for reading and writing
Excel, Word, and PDF documents in Windows store apps.
</div>
<h3>
<a href="#">Metro Studio</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Syncfusion Metro Studio is a collection of over 2500 Metro-style icon
templates that can be easily customized to create thousands of unique Metro
icons.
</div>
</div>
```

## JAVASCRIPT

```
// Configure multiple item selection for Accordion
var sample = new ej.Accordion($("#accordion"), {
  selectedItems: [0, 2],
  enableMultipleOpen: true
});
```

Output for Accordion control with the multiple selected items is as follows.



### AJAX settings

**Accordion** widgets allow you to load content for the **Accordion** panel using **AJAX**. This renders content from the specified **URL** location that is set to the anchor tag. You can set the destination file URL string by using **ajaxUrl** property **AJAX** contents enables you to load the content of the **Accordion** panel when it is expanded. This enhances the **Accordion** control efficiency when a large content is to be loaded into the panel.

### Populate accordion with AJAX content

- Create HTML files with required content to be loaded for Accordion panel and save it in your drive location.
- In the HTML page, define a div element that is a container for Accordion widget and add the contents URL location in the <a> tag accordingly.

### HTML

```
<div id="ajaxAccordion" style="width: 650px">
<h3>
<a href="mvccontent.html">Model-view-controller (MVC) </a>
</h3>
<div>
</div>
<h3>
<a href="wpfcontent.html">WPF
</h3>
<div>
</div>
<h3>
<a href="#">WCF</a>
</h3>
<div>
<p>
```



WCF is a tool often used to implement and deploy a service-oriented architecture (SOA). It is designed using service-oriented architecture principles to support distributed computing where services have remote consumers. Clients can consume multiple services; services can be consumed by multiple clients. Services are loosely coupled to each other. Services typically have a WSDL interface (Web Services Description Language) that any WCF client can use to consume the service, regardless of which platform the service is hosted on. WCF implements many advanced Web services (WS) standards such as WS-Addressing, WS-ReliableMessaging and WS-Security. With the release of .NET Framework 4.0, WCF also provides RSS Syndication Services, WS-Discovery, routing and better support for REST services.

</p>  
</div>  
</div>

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AccordionComponent {
  $(function () {
    var sample = new ej.Accordion($("#ajaxAccordion"), {
    });
  });
}
```

Output for Accordion control with loaded AJAX content is as follows.

### – Model–view–controller (MVC)

Model–view–controller (MVC) is a software architecture pattern which separates the representation of information from the user's interaction with it. The model consists of application data, business rules, logic, and functions. A view can be any output representation of data, such as a chart or a diagram. Multiple views of the same data are possible, such as a bar chart for management and a tabular view for accountants. The controller mediates input, converting it to commands for the model or view. The central ideas behind MVC are code reusability and in addition to dividing the application into three kinds of components, the MVC design defines the interactions between them.

- **A controller** can send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document). It can also send commands to the model to update the model's state (e.g., editing a document).
- **A model** notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. A passive implementation of MVC omits these notifications, because the application does not require them or the software platform does not support them.
- **A view** requests from the model the information that it needs to generate an output representation to the user.

+ WPF

+ WCF

### [View multiple contents](#)

By default **Accordion** allows only one panel to be in expanded state. You can enable multiple panels in expand state by setting **enableMultipleOpen** to **true**.

### Enabling multiple panel open

The following steps explain you to enable multiple panel for **Accordion**.

In an HTML page, define a div element that is a container for **Accordion** widget and add the contents correspondingly

### HTML

```
<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Orubase is the only mobile application development Framework built
especially for developing complex line-of-business mobile applications
targeting iOS, Android, and Windows Phone platforms in the shortest possible
time frame.
</div>
```

```

<h3>
<a href="#">WinRTXAML</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Essential Studio for WinRT contains all the controls you need to build line-
of-business tablet applications including grid, chart, map, tree map, SSRS
report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and
much more. It also includes a unique set of controls for reading and writing
Excel, Word, and PDF documents in Windows store apps.
</div>
<h3>
<a href="#">Metro Studio</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Syncfusion Metro Studio is a collection of over 2500 Metro-style icon
templates that can be easily customized to create thousands of unique Metro
icons.
</div>
</div>

```

## JAVASCRIPT

```

// Enable multiple open for Accordion
$("#accordion").ejAccordion({
  enableMultipleOpen: true
});

```

Following screenshot is the output for Accordion control on enableMultipleOpen set to true.



## Accordion Panel enabler

### Enable/Disable widget

**You can enable or disable the Accordion** widget on initial rendering using the **enabled** property. By default **enabled** property is set to true and the **Accordion** panels are active always.

The following steps explain you on how to disable the **Accordion** widget

In an HTML page, define a <div> element that is a container for Accordion widget and add the contents correspondingly

## HTML

```

<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Orubase is the only mobile application development Framework built
especially for developing complex line-of-business mobile applications
targeting iOS, Android, and Windows Phone platforms in the shortest possible
time frame.
</div>

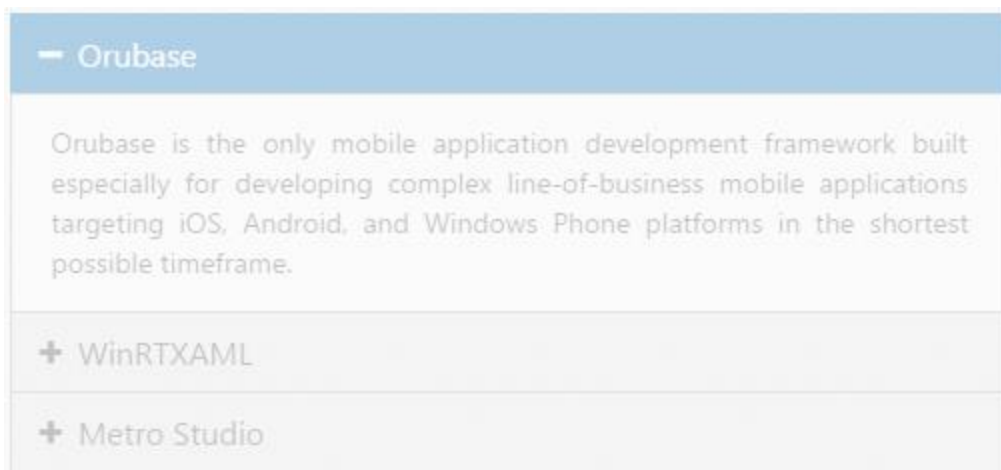
```

```
<h3>
<a href="#">WinRTXAML</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Essential Studio for WinRT contains all the controls you need to build line-
of-business tablet applications including grid, chart, map, tree map, SSRS
report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and
much more. It also includes a unique set of controls for reading and writing
Excel, Word, and PDF documents in Windows store apps.
</div>
<h3>
<a href="#">Metro Studio</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Syncfusion Metro Studio is a collection of over 2500 Metro-style icon
templates that can be easily customized to create thousands of unique Metro
icons.
</div>
</div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AccordionComponent {
$(function () {
// To disable the control set enabled property value as false
var sample = new ej.Accordion($("#accordion"), {
enabled: false
});
});
}
```

Output for disabled Accordion control is as follows.



### Enable panel items

You can enable the **Accordion** widget items on initial loading using **enabledItems** property. This property takes array of indices whose panel needs to be enabled in **Accordion** widget.

The **disabledItems** property disables the **Accordion** items based on the index. This takes array of indices whose panel is to be disabled.

### Enabling accordion panel items

The following steps explain you on how to enable the panel items in **Accordion** widget

In an HTML page, define a <div> element that is a container for Accordion widget and add the contents correspondingly

### HTML

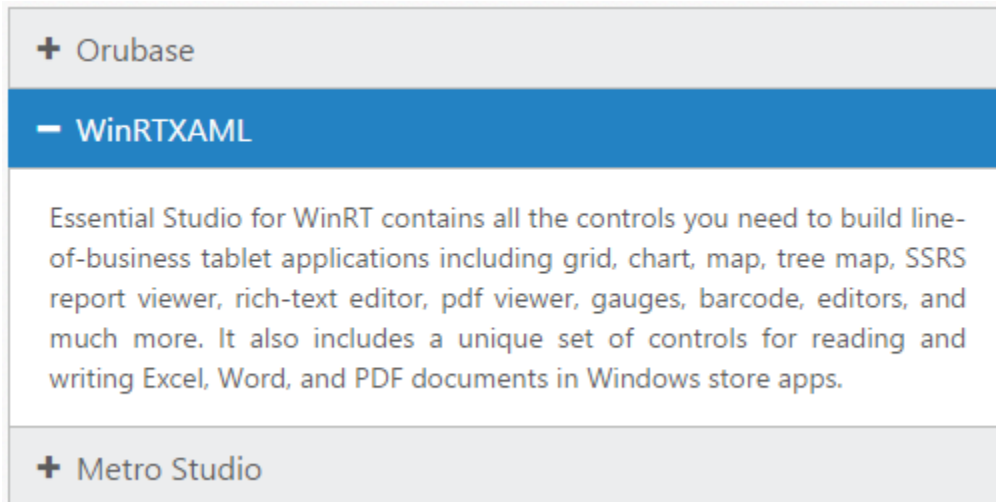
```
<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Orubase is the only mobile application development Framework built
especially for developing complex line-of-business mobile applications
targeting iOS, Android, and Windows Phone platforms in the shortest possible
time frame.
</div>
<h3>
<a href="#">WinRTXAML</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Essential Studio for WinRT contains all the controls you need to build line-
of-business tablet applications including grid, chart, map, tree map, SSRS
report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and
much more. It also includes a unique set of controls for reading and writing
Excel, Word, and PDF documents in Windows store apps.
</div>
<h3>
<a href="#">Metro Studio</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Syncfusion Metro Studio is a collection of over 2500 Metro-style icon
templates that can be easily customized to create thousands of unique Metro
icons.
</div>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AccordionComponent {
$(function () {
var sample = new ej.Accordion($("#accordion"), {
enabledItems: [1, 2],
```

```
disabledItems: [0],
selectedItemIndex: 1,
enableMultipleOpen: true
});
});
}
```

Output for Accordion control with some enabled and disabled items, where first panel is disabled and it can't be expanded or collapsed is as follows.



## State Persistence

**Accordion** widget can store the model value in the browser cookies and on every time after initial rendering, the control get the model from the cookie only. Using **enablePersistence** property you can store the model value in cookies. Thus when any changes are made dynamically then those values are updated in cookie. On refreshing the page the past state of the **Accordion** control is maintained in cookie and control is rendered from it.

### Configure state persistence of Accordion panel

The following steps explains to enable state maintenance for **Accordion**.

In an HTML page, define a div element that is a container for Accordion widget and add the contents correspondingly

### HTML

```
<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Orubase is the only mobile application development Framework built
especially for developing complex line-of-business mobile applications
targeting iOS, Android, and Windows Phone platforms in the shortest possible
time frame.
</div>
<h3>
<a href="#">WinRTXAML</a>
```

```

</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Essential Studio for WinRT contains all the controls you need to build line-
of-business tablet applications including grid, chart, map, tree map, SSRS
report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and
much more. It also includes a unique set of controls for reading and writing
Excel, Word, and PDF documents in Windows store apps.
</div>
<h3>
<a href="#">Metro Studio</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Syncfusion Metro Studio is a collection of over 2500 Metro-style icon
templates that can be easily customized to create thousands of unique Metro
icons.
</div>
</div>

```

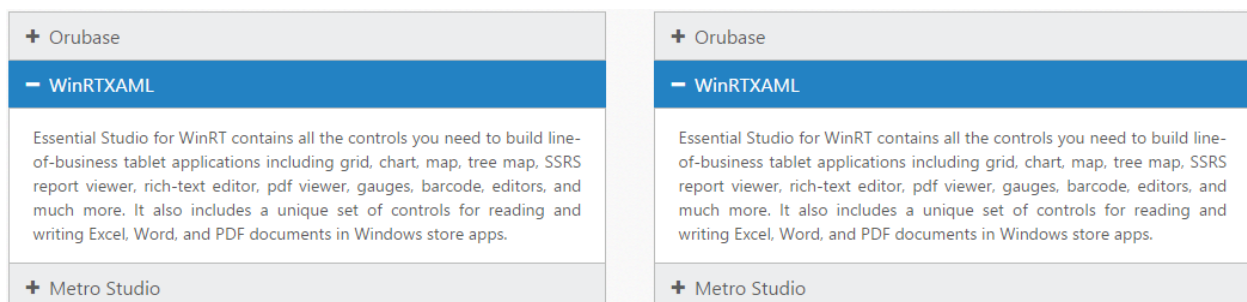
## JAVASCRIPT

```

// Configure enablePersistence for Accordion
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AccordionComponent {
$(function () {
var sample = new ej.Accordion($("#accordion"), {
enablePersistence: true
});
});
}

```

Output after page refresh maintaining the previous state of Accordion widget is as follows.



## Appearance and Styling

### Adjusting Accordion size

You can customize the **Accordion** panel height using **heightAdjustMode** property. It can be set to **enum** values like **content**, **fill** or **auto**. By default **heightAdjustMode** is set to **content** so the panel height is adjusted to the content size.

### Configure Height of Accordion panel

The following steps explain you on how to configure **Accordion** panel height.

In an HTML page, define a <div> element that is a container for Accordion widget and add the contents correspondingly

### HTML

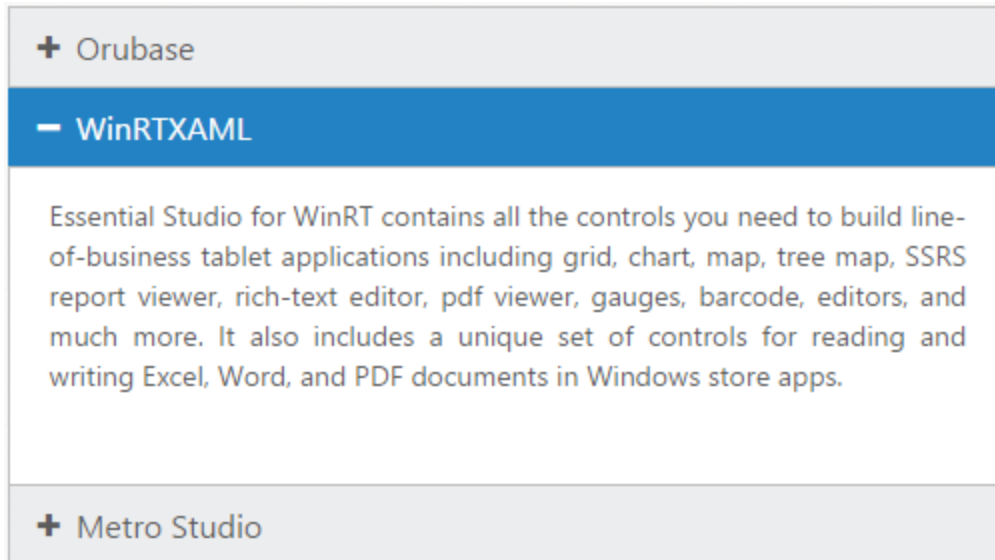
```
<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Orubase is the only mobile application development Framework built
especially for developing complex line-of-business mobile applications
targeting iOS, Android, and Windows Phone platforms in the shortest possible
time frame.
</div>
<h3>
<a href="#">WinRTXAML</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Essential Studio for WinRT contains all the controls you need to build line-
of-business tablet applications including grid, chart, map, tree map, SSRS
report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and
much more. It also includes a unique set of controls for reading and writing
Excel, Word, and PDF documents in Windows store apps.
</div>
<h3>
<a href="#">Metro Studio</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Syncfusion Metro Studio is a collection of over 2500 Metro-style icon
templates that can be easily customized to create thousands of unique Metro
icons.
</div>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AccordionComponent {
$(function () {
var sample = new ej.Accordion($("#accordion"), {
heightAdjustMode: "auto"
});
});
}
```

Output for Accordion control when panel height is set to auto so that the maximum content height and Fill for minimum content height in all the panels is as follows.





### Rounded corner

You can customize the shape of the **Accordion** widget from regular rectangular shape to rounded rectangle shape enabling **roundedCorner** property that is set to false by default.

### Enabling Rounded corner property

The following steps explain you in enabling the **showRoundedCorner** property for an **Accordion** control.

In an HTML page, define a <div> element that is a container for Accordion widget and add the contents correspondingly

### HTML

```
<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Orubase is the only mobile application development Framework built
especially for developing complex line-of-business mobile applications
targeting iOS, Android, and Windows Phone platforms in the shortest possible
time frame.
</div>
<h3>
<a href="#">WinRTXAML</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Essential Studio for WinRT contains all the controls you need to build line-
of-business tablet applications including grid, chart, map, tree map, SSRS
report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and
much more. It also includes a unique set of controls for reading and writing
Excel, Word, and PDF documents in Windows store apps.
</div>
<h3>
<a href="#">Metro Studio</a>
```

```

</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Syncfusion Metro Studio is a collection of over 2500 Metro-style icon
templates that can be easily customized to create thousands of unique Metro
icons.
</div>
</div>

```

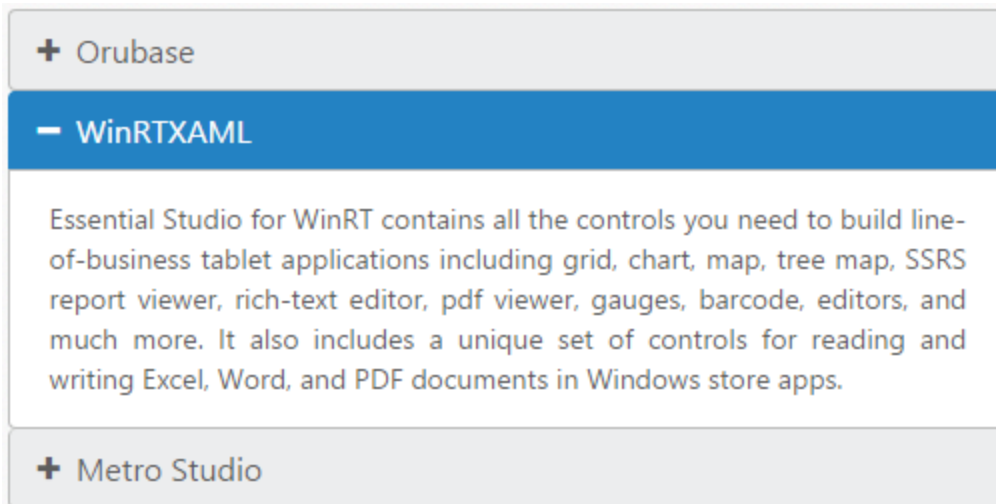
## JAVASCRIPT

```

// Enable showRoundedCorner for Accordion
var sample = new ej.Accordion($("#accordion"), {
  showRoundedCorner: true
});

```

Output for accordion widget when “showRoundedCorner” is set to “true” is as follows.



### Customize Accordion icon

**Accordion** widget allows you to customize the icons using **customIcon** option that has two properties **header** and **selectedHeader**. By default, the classes of header and selectedHeader are e-collapse and e-expand respectively. By setting the desired CSS class names for these properties as required overrides the default icons with customized icons.

### Configuring custom icon for Accordion

The following steps explain you the configuration of icon for an **Accordion** control.

In an HTML page, define a <div> element that is a container for Accordion widget and add the contents correspondingly

## HTML

```

<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->

```

Orubase is the only mobile application development Framework built especially for developing complex line-of-business mobile applications targeting iOS, Android, and Windows Phone platforms in the shortest possible time frame.

```
</div>
```

```
<h3>
```

```
<a href="#">WinRTXAML</a>
```

```
</h3>
```

```
<div>
```

```
<!-- add accordion contents here to load contents under this header -->
```

Essential Studio for WinRT contains all the controls you need to build line-of-business tablet applications including grid, chart, map, tree map, SSRS report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and much more. It also includes a unique set of controls for reading and writing Excel, Word, and PDF documents in Windows store apps.

```
</div>
```

```
<h3>
```

```
<a href="#">Metro Studio</a>
```

```
</h3>
```

```
<div>
```

```
<!-- add accordion contents here to load contents under this header -->
```

Syncfusion Metro Studio is a collection of over 2500 Metro-style icon templates that can be easily customized to create thousands of unique Metro icons.

```
</div>
```

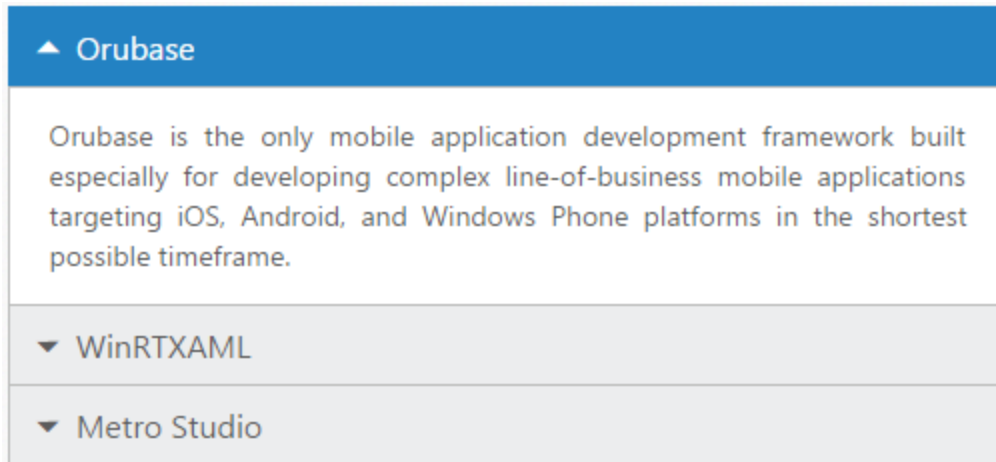
```
</div>
```

## JAVASCRIPT

```
// Set the "e-arrowheaddown" and "e-arrowheadup" classes to header and
selectedHeader properties. "e-arrowheaddown" and "e-arrowheadup" are
available in ej.widgets.core.min.css file.
```

```
var sample = new ej.Accordion($("#accordion"), {
  customIcon: {
    /* To set icon for the collapsed accordion headers */
    header: "e-arrowheaddown",
    /* To set icon for the selected accordion headers */
    selectedHeader: "e-arrowheadup"
  }
});
```

Output for Accordion widget with customized icons is as follows.



## Animations

### Set animation

By default the **Animation** for expanding and collapsing is enabled. To remove the Animation you can set the **enableAnimation** property to **false**. This restricts customizing animations as well. By default **enableAnimation** is set to **true**.

Following code disables **Animation** for **Accordion**.

### JAVASCRIPT

```
var sample = new ej.Accordion($("#accordion"), {
  enableAnimation: false
});
```

### Expand and collapse speed

This feature allows you to set the speed for expanding and collapsing the **Accordion** panels. By default it is set to 300 in milliseconds. By configuring the animation speed you can optimize the delay in loading the panel content.

The following code sample sets value for **expandSpeed** and **collapseSpeed** properties,

### JAVASCRIPT

```
var sample = new ej.Accordion($("#accordion"), {
  expandSpeed: 600,
  collapseSpeed: 1000,
  collapsible: true
});
```

## Theme

You can control the style and appearance of Accordion control based on **CSS** classes. In order to apply styles to the Accordion widget, you can refer two files, **ej.widgets.core.min.css** and **ej.theme.min.css**. When you refer **ej.widgets.all.min.css** file, then it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.widgets.all.min.css** is the combination of these two.

By default, there are 16 theme support available for RadialMenu component, namely:

- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap
- high-contrast-01
- high-contrast-02
- material
- office-365

### CSS class

**CSS** class can be used to customize the **Accordion** control appearance. Define a **CSS** class as you're your requirement and assign the class name to **cssClass** property.

### Configure AutoComplete textbox using CSS class

The following steps allows you to configure **CSS** class for an **Accordion** widget.

In the HTML page, define a <div> element that is a container for Accordion widget and add the contents correspondingly

### HTML

```
<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Orubase is the only mobile application development Framework built
especially for developing complex line-of-business mobile applications
targeting iOS, Android, and Windows Phone platforms in the shortest possible
time frame.
</div>
<h3>
<a href="#">WinRTXAML</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Essential Studio for WinRT contains all the controls you need to build line-
of-business tablet applications including grid, chart, map, tree map, SSRS
report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and
much more. It also includes a unique set of controls for reading and writing
Excel, Word, and PDF documents in Windows store apps.
</div>
<h3>
<a href="#">Metro Studio</a>
```

```
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Syncfusion Metro Studio is a collection of over 2500 Metro-style icon
templates that can be easily customized to create thousands of unique Metro
icons.
</div>
</div>
```

## JAVASCRIPT

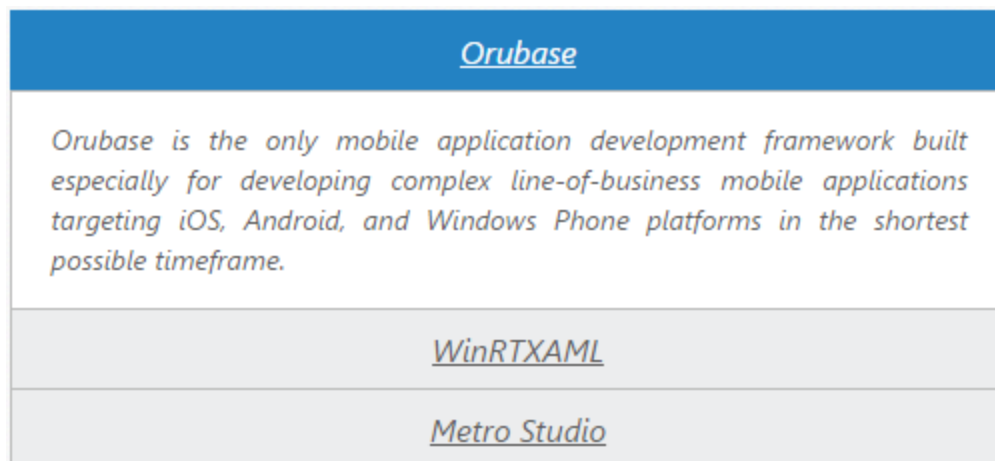
```
// Set the cssClass property for Accordion
var sample = new ej.Accordion($("#accordion"), {
  cssClass: "customCss"
});
```

Define CSS class for customizing the Accordion.

## CSS

```
<style class="cssStyles">
.customCss
{
font-style: italic;
text-align:justify;
}
.customCss span.e-icon {
display: none !important;
}
.customCss h3
{
text-decoration:underline;
text-align:center;
}
</style>
```

Output for Accordion with customized CSS property to hide the Accordion icon and format its content is as follows.



## RTL Support

This feature supports to change the left-to-right alignment of the **Accordion** widget to right-to-left (RTL).

### Enabling RTL Support

The following steps explains you in enabling the right-to-left property for an **Accordion**.

In an HTML page, define a div element that is a container for Accordion widget and add the contents correspondingly

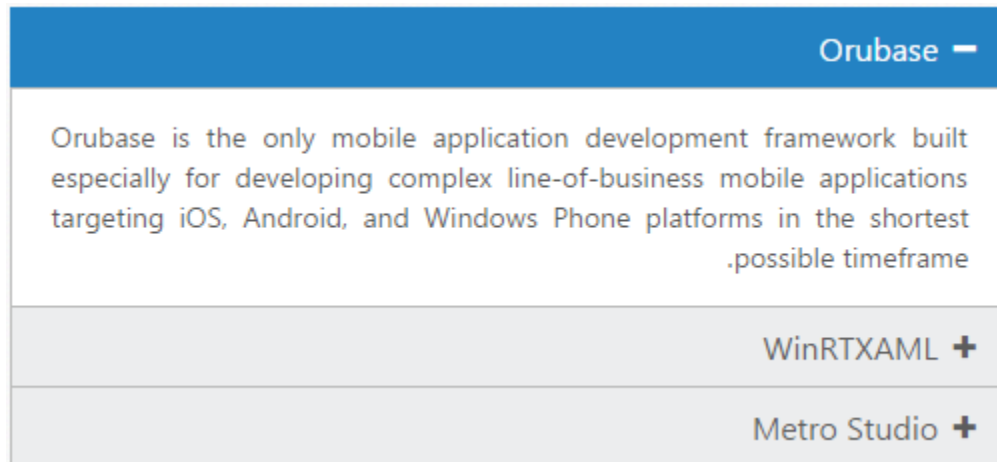
### HTML

```
<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Orubase is the only mobile application development Framework built
especially for developing complex line-of-business mobile applications
targeting iOS, Android, and Windows Phone platforms in the shortest possible
time frame.
</div>
<h3>
<a href="#">WinRTXAML</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Essential Studio for WinRT contains all the controls you need to build line-
of-business tablet applications including grid, chart, map, tree map, SSRS
report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and
much more. It also includes a unique set of controls for reading and writing
Excel, Word, and PDF documents in Windows store apps.
</div>
<h3>
<a href="#">Metro Studio</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Syncfusion Metro Studio is a collection of over 2500 Metro-style icon
templates that can be easily customized to create thousands of unique Metro
icons.
</div>
</div>
```

### JAVASCRIPT

```
// Enable RTL for accordion control
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AccordionComponent {
$(function () {
var sample = new ej.Accordion($("#accordion"), {
enableRTL: true
});
});
}
```

Output for accordion when “enableRTL” is set to “true” is as follows,



## Keyboard Navigation

You can use **Keyboard** shortcut keys as an alternative to the mouse on using Accordion widget using **allowKeyboardNavigation** property. However you will have to focus the control to enable the keyboard navigation. Accordion Widget allows you to perform all kind of actions using keyboard shortcuts.

List of Keyboard shortcut keys

| Shortcut Key   | Description                        |
|----------------|------------------------------------|
| Access key + j | Focuses into the accordion control |
| Up             | Moves to previous panel            |
| Down           | Moves to next panel                |
| Left           | Moves to previous panel            |
| Right          | Moves to next panel                |
| Home           | Moves to the first accordion panel |
| End            | Moves to the last accordion panel  |

## Configure keyboard interaction

The following steps explains you on how to enable keyboard interaction for an **Accordion** widget.

In an HTML page, define a <div> element that is a container for Accordion widget and add the contents correspondingly

## HTML

```
<div id="accordion" style="width: 500px">
<h3>
<a href="#">Orubase</a>
</h3>
<div>
```



```

<!-- add accordion contents here to load contents under this header -->
Orubase is the only mobile application development Framework built
especially for developing complex line-of-business mobile applications
targeting iOS, Android, and Windows Phone platforms in the shortest possible
time frame.
</div>
<h3>
<a href="#">WinRTXAML</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Essential Studio for WinRT contains all the controls you need to build line-
of-business tablet applications including grid, chart, map, tree map, SSRS
report viewer, rich-text editor, pdf viewer, gauges, barcode, editors, and
much more. It also includes a unique set of controls for reading and writing
Excel, Word, and PDF documents in Windows store apps.
</div>
<h3>
<a href="#">Metro Studio</a>
</h3>
<div>
<!-- add accordion contents here to load contents under this header -->
Syncfusion Metro Studio is a collection of over 2500 Metro-style icon
templates that can be easily customized to create thousands of unique Metro
icons.
</div>
</div>

```

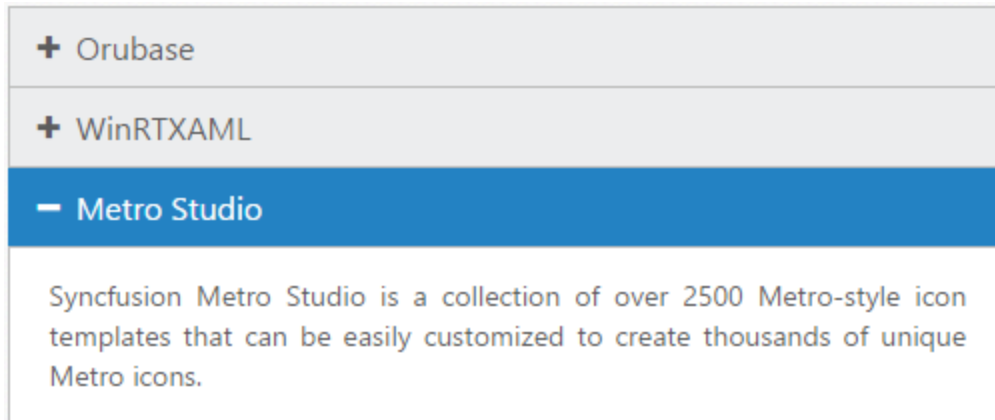
## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AccordionComponent {
$(function () {
// Configure Keyboard navigation for Accordion by setting
allowKeyboardNavigation property to true. Now define the script to focus the
Accordion widget on AccessKey + J
var sample = new ej.Accordion($("#accordion"), {
allowKeyboardNavigation: true
});
// Define script to focus into the Accordion control on Alt + J shortcut
keys.
$(document).on("keydown", function (e) {
if (e.altKey && e.keyCode === 74) { // j- key code.
$("#accordion").focus();
}
});
});
}

```

Output for Accordion widget focused and navigated to last item using Keyboard navigation is as follows.



## Template Support

In **JavaScript**, you can load the contents or **HTML** elements directly inside the **<div>** element that you are going to convert as **Accordion** control.

### HTML

```
<div id="pizzaMenu" style="width: 500px">
<h3>
<a href="#">GARDEN FRESH (Veg)</a>
</h3>
<div>

<div class="ingredients">
Rate      : $50
<br />
Ingredients : cheese, onions, green capsicums & tomatoes.
</div>
</div>
<h3>
<a href="#">CORN & SPINACH (Veg)</a>
</h3>
<div>

<div class="ingredients">
Rate      : $70
<br />
Ingredients : cheese, sweet corn & green capsicums.
</div>
</div>
<h3>
<a href="#">CHICKEN DELITE (Non-veg)</a>
</h3>
<div>

<div class="ingredients">
Rate      : $100
<br />
Ingredients : cheese, chicken chunks, onions & pineapple chunks.
</div>
</div>
</div>
```

**JAVASCRIPT**

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module AccordionComponent {
    $(function () {
        var sample = new ej.Accordion($("#pizzaMenu"), {
            enableMultipleOpen: true
        });
    });
}
```

**— GARDEN FRESH (Veg)**

Rate : \$50

Ingredients : cheese, onions, green capsicums & tomatoes.

**— CORN & SPINACH (Veg)**

Rate : \$70

Ingredients : cheese, sweet corn & green capsicums.

**— CHICKEN DELITE (Non-veg)**

Rate : \$100

Ingredients : cheese, chicken chunks, onions & pineapple chunks.

## Autocomplete

### Overview

The **Typescript** AutoComplete component is a textbox control that provides a list of suggestions based on your query. When you enter a text into the text box, the control performs a search operation and provides a list of results. There are several filter types available to perform the search.

### Key Features

- Multi word Search: Supports searching using multiple words
- Data binding: Supports data binding with JSON data and remote data.
- Template: Supports templates for the suggestion list content.
- Highlight Search: Supports highlighting the typed text in the suggestion list.
- Auto Fill: Allows the control to automatically select the first suggested item.
- Keyboard Interaction: Supports using the keyboard to focus on the control, navigate through the suggestion results, and select a value.

### Getting Started

Using the following steps, you can create a **Typescript** AutoComplete component. The basic rendering of **Typescript** AutoComplete is achieved with default functionality.

#### Creating an AutoComplete in Typescript

You can create a **Typescript** application with the help of the given [Typescript Getting Started Documentation](#).

Within an index.html file and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Typescript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.web
.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add AutoComplete here-->
</body>
</html>
```

The AutoComplete can be created from a HTML 'input' element with the HTML 'id' attribute and pre-defined options set to it. To create the AutoComplete, you should call the `ejAutocomplete` jQuery plug-in function.

#### HTML

```
<input id="autocomplete" />
<script src="app.js"></script>
```

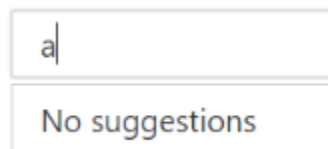
Create app.ts file and past the below content

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module TabComponent {
$(function () {
let sample = new ej.Autocomplete($("#autocomplete"));
});
}
```

Now build your application, so that the **app.js** file is automatically generated and got added to your project (User have nothing to do with this file). Now, whatever code changes that you make in **app.ts** file will be reflected in app.js file automatically.

This will render an Autocomplete with no suggestion on executing.



### Data Binding

The data for AutoComplete suggestion list which can be populated using the dataSource property.

### JS

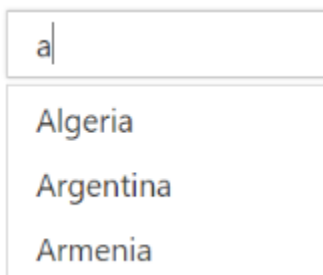
```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module TabComponent {
$(function () {
let data = [
{ text: "Algeria", sprite: "flag-dz" }, { text: "Argentina", sprite: "flag-ar" },
{ text: "Armenia", sprite: "flag-am" }, { text: "Brazil", sprite: "flag-br" },
{ text: "Bangladesh", sprite: "flag-bd" }, { text: "Canada", sprite: "flag-ca" },
{ text: "Cuba", sprite: "flag-cu" }, { text: "China", sprite: "flag-cn" },
{ text: "Denmark", sprite: "flag-dk" }, { text: "Estonia", sprite: "flag-ee" },
{ text: "Egypt", sprite: "flag-eg" }, { text: "France", sprite: "flag-fr" },
{ text: "Finland", sprite: "flag-fi" }, { text: "Greenland", sprite: "flag-gl" },
{ text: "India", sprite: "flag-in" }, { text: "Indonesia", sprite: "flag-id" },
{ text: "Malaysia", sprite: "flag-my" }, { text: "Mexico", sprite: "flag-mx" },
{ text: "New Zealand", sprite: "flag-nz" }, { text: "Netherlands", sprite: "flag-nl" },
];
}
```

```

{ text: "Norway", sprite: "flag-no" }, { text: "Portugal", sprite: "flag-pt"
},
{ text: "Poland", sprite: "flag-pl" }, { text: "Qatar", sprite: "flag-qa" },
{ text: "Romania", sprite: "flag-ro" }, { text: "Spain", sprite: "flag-es"
},
{ text: "Singapore", sprite: "flag-sg" }, { text: "Saudi Arabia", sprite:
"flag-sa" },
{ text: "Thailand", sprite: "flag-th" }, { text: "Turkey", sprite: "flag-tr"
},
{ text: "Ukraine", sprite: "flag-ua" }, { text: "United States", sprite:
"flag-us" },
{ text: "Uruguay", sprite: "flag-uy" }, { text: "Viet Nam", sprite: "flag-
vn" },
{ text: "Yemen", sprite: "flag-ye" }
];
let sample = new ej.Autocomplete($("#autocomplete"), {watermarkText:"Select
a country", dataSource: data, fields: { text: "text", key: "sprite" } });
});
}

```

Run the above code to render the following output:



### Enable Popup Button

We can enable the popup button of AutoComplete by using showPopupButton property which helps you to show all the available suggestions on clicking it.

### JS

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module TabComponent {
$(function () {
let data = [
{ text: "Algeria", sprite: "flag-dz" }, { text: "Argentina", sprite: "flag-
ar" },
{ text: "Armenia", sprite: "flag-am" }, { text: "Brazil", sprite: "flag-br"
},
{ text: "Bangladesh", sprite: "flag-bd" }, { text: "Canada", sprite: "flag-
ca" },
{ text: "Cuba", sprite: "flag-cu" }, { text: "China", sprite: "flag-cn" },
{ text: "Denmark", sprite: "flag-dk" }, { text: "Estonia", sprite: "flag-ee"
},
{ text: "Egypt", sprite: "flag-eg" }, { text: "France", sprite: "flag-fr" },
{ text: "Finland", sprite: "flag-fi" }, { text: "Greenland", sprite: "flag-
gl" },

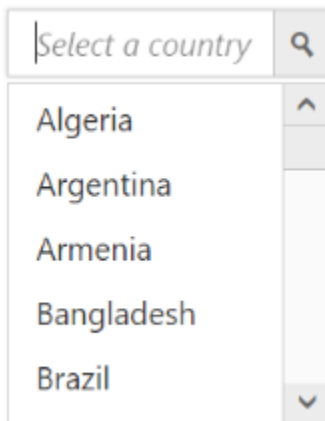
```

```

{ text: "India", sprite: "flag-in" }, { text: "Indonesia", sprite: "flag-id"
},
{ text: "Malaysia", sprite: "flag-my" }, { text: "Mexico", sprite: "flag-mx"
},
{ text: "New Zealand", sprite: "flag-nz" }, { text: "Netherlands", sprite:
"flag-nl" },
{ text: "Norway", sprite: "flag-no" }, { text: "Portugal", sprite: "flag-pt"
},
{ text: "Poland", sprite: "flag-pl" }, { text: "Qatar", sprite: "flag-qa" },
{ text: "Romania", sprite: "flag-ro" }, { text: "Spain", sprite: "flag-es"
},
{ text: "Singapore", sprite: "flag-sg" }, { text: "Saudi Arabia", sprite:
"flag-sa" },
{ text: "Thailand", sprite: "flag-th" }, { text: "Turkey", sprite: "flag-tr"
},
{ text: "Ukraine", sprite: "flag-ua" }, { text: "United States", sprite:
"flag-us" },
{ text: "Uruguay", sprite: "flag-uy" }, { text: "Viet Nam", sprite: "flag-
vn" },
{ text: "Yemen", sprite: "flag-ye" }
];
let sample = new ej.Autocomplete($("#autocomplete"), {showPopupButton:true,
watermarkText:"Select a country", dataSource: data, fields: { text: "text",
key: "sprite" } });
});
}

```

Run the above code to render the following output:




---

*Note: You can find the Autocomplete properties from the [API reference](#) document*

---

## Data Binding

In order to render the Autocomplete widget, the data needs to be bound to it in a proper way. The below sections explain about binding local or remote data to the Autocomplete widget.

### Fields

The Autocomplete widget has a field property (object) which holds the properties to map with dataSource fields. Whenever we bind a data to Autocomplete, corresponding fields must be mapped using this property.

The field object contains the following properties.

| Fields property       | Description   |
|-----------------------|---|
| <u>text</u>           | Maps the display text in the suggestion list          |
| <u>key</u>            | Maps to the unique key field in the data source       |
| <u>groupBy</u>        | Allows to enable grouping based on the assigned field |
| <u>htmlAttributes</u> | Maps to set the HTML attribute for the list items     |

#### Local data

The local data must be an array of JSON objects which is assigned for the Autocomplete widget `dataSource` property.

In the below example name and index fields are mapped with text and key properties of the field object respectively.

#### HTML

```
<input type="text" id="autocomplete" />
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AutocompleteComponent {
    var countriesField = [
        { name: "Austria", index: "C1" },
        { name: "Australia", index: "C2" }, { name: "Antarctica", index: "C3" },
        { name: "Bangladesh", index: "C4" }, { name: "Belgium", index: "C5" },
        { name: "Brazil", index: "C6" },
        { name: "Canada", index: "C7" }, { name: "China", index: "C8" },
        { name: "Cuba", index: "C9" },
        { name: "Denmark", index: "C10" }, { name: "Dominica", index: "C11" },
        { name: "Europe", index: "C12" }, { name: "Egypt", index: "C13" },
        { name: "England", index: "C14" },
        { name: "India", index: "C15" }, { name: "Indonesia", index: "C16" }
    ];
    $(function () {
        var autocompleteInstance = new ej.Autocomplete($("#autocomplete"), {
            dataSource: countriesField,
            fields: { key: "index", text: "name" },
            width: 205
        });
    });
}
```



Antarctica  
 Australia  
 Austria

### Remote data

#### OData

[OData](#) is a standardized protocol for creating and consuming the data. You can retrieve data from OData service by using [ej.DataManager](#) and the queries can be added using [ej.Query\(\)](#).

Here ContactName and SupplierID fields are mapped with text and key properties respective to the field object.

#### HTML

```
<input type="text" id="autocomplete" />
<script type="text/javascript">
  /* Create DataManager */
  var dataManger = ej.DataManager({
    /* OData service */
    url: "http://mvc.syncfusion.com/Services/Northwnd.svc/"
  });
  /* Create Query */
  var query = ej.Query().from("Suppliers").select("SupplierID",
    "ContactName");
  var autocompleteInstance = new ej.Autocomplete($("#autocomplete"), {
    dataSource: dataManger,
    query: query,
    fields: { text: "ContactName", key: "SupplierID" },
    width: 205
  });
</script>
```

Anne Heikkonen  
 Antonio del Valle Saavedra

### Multiple Columns

The Autocomplete adds support for selecting multiple columns in the dropdown list. This multiple column options can be enabled and customized through the [MultiColumnSettings](#) property.

In AutoComplete Multiple Column search is based on [searchColumnIndices](#) property which allows user to search text for any number of fields in the suggestion list without modifying the selected text format.

In AutoComplete Multiple Column searched value is updated to autocomplete input box based on [stringFormat](#) property which specifies column indices values to be updated.

| Name  | Description  |
|---|--|
| multiColumnSettings.enable                  | Allow list of data to be displayed in several columns.                                 |
| multiColumnSettings.showHeader              | Allow header text to be displayed in corresponding columns.                            |
| multiColumnSettings.stringFormat            | Specifies the format for displaying a selected value and columns to be searched.       |
| multiColumnSettings.columns                 | Field and Header Text collections can be defined and customized through this field.    |
| multiColumnSettings.columns.field           | Field to be mapped from the dataSource to the columns in Autocomplete suggestion list. |
| multiColumnSettings.columns.headerText      | Get or set a value that indicates to display the title of that particular column.      |
| multiColumnSettings.columns.cssClass        | Field to be mapped for to add user defined CSS class for the specific column.          |
| multiColumnSettings.columns.type            | Get or set the data type for a individual column.                                      |
| multiColumnSettings.columns.filterType      | Get or set the search filter type for the column.                                      |
| multiColumnSettings.columns.headerTextAlign | Used to align or position the header value in the column.                              |
| multiColumnSettings.columns.textAlign       | Used to align or position all the values in a column.                                  |

## Example

### HTML

```
<input type="text" id="autocomplete" />
/* Local Data */
var carList = [
{ "EmployeeID": 1, "FirstName": "Nancy", "City": "Seattle" },
{ "EmployeeID": 10, "FirstName": "Laura", "City": "Seattle" },
{ "EmployeeID": 11, "FirstName": "Janet", "City": "Kirkland" },
{ "EmployeeID": 12, "FirstName": "Michael", "City": "London" },
{ "EmployeeID": 13, "FirstName": "Steven", "City": "London" },
{ "EmployeeID": 14, "FirstName": "Andrew", "City": "Tacoma" },
{ "EmployeeID": 15, "FirstName": "Robert", "City": "London" },
{ "EmployeeID": 16, "FirstName": "Margaret", "City": "Redmond" },
{ "EmployeeID": 17, "FirstName": "Steven", "City": "London" },
{ "EmployeeID": 18, "FirstName": "Michael", "City": "London" },
{ "EmployeeID": 19, "FirstName": "Robert", "City": "London" }, ];
var autocompleteInstance = new ej.Autocomplete($("#selectCar"), {
dataSource: carList,
width: 400,
multiColumnSettings:{
```

```
enable:true,
showHeader:true,
stringFormat:"{1}",
searchColumnIndices[0,1,2],
columns:[
{"field": "EmployeeID" , "headerText":"EmployeeID"},
{"field": "FirstName" , "headerText":"FirstName"},
{"field": "City" , "headerText":"City"}
]
}
});
```

**Note:** Here [stringFormat](#) is "{0} {1} {2}" so the search will be based on column indices 0, 1 and 2.

| 1          |           |          |
|------------|-----------|----------|
| EmployeeID | FirstName | City     |
| 11         | Janet     | Kirkland |
| 12         | Michael   | London   |
| 18         | Michael   | London   |
| 15         | Robert    | London   |

## MultiSelection

The AutoComplete widget helps you to select multiple values from the suggestion list using the multiSelect property.

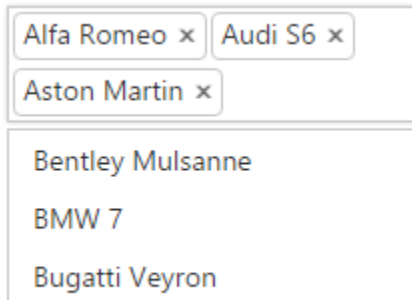
There are two types of multi-selection mode.

1. VisualMode – Selection values are displayed in separate box with close like button.
2. Delimiter – Selection values are separated using the delimiter character which can be specified using delimiterChar API.

## HTML

```
<input type="text" id="autocomplete" />
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AutocompleteComponent{
/* Local Data */
var carList = [
"Audi S6", "Audi S6", "Austin-Healey", "Alfa Romeo", "Aston Martin",
"BMW 7 ", "Bentley Mulsanne", "Bugatti Veyron",
"Chevrolet Camaro", "Cadillac ",
"Duesenberg J ", "Dodge Sprinter",
"Elantra", "Excavator",
"Ford Boss 302", "Ferrari 360", "Ford Thunderbird ",
"GAZ Siber"];
var autocompleteInstance =new ej.Autocomplete($("#autocomplete"), {
dataSource: carList,
```

```
width: 205,
multiSelectMode:ej.MultiSelectMode.VisualMode
});
</script>
```



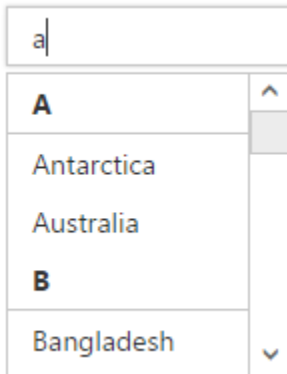
## Grouping

The suggestion list items can be grouped by providing a header for each set of items. The grouping will be defined based on the “groupBy” API in fields object.

Here the category field is mapped with groupBy field.

## HTML

```
<input type="text" id="autocomplete" />
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AutocompleteComponent{
/* Local Data */
var countries = [
{ text: "Australia", category: "A" }, { text: "Antarctica", category: "A" },
{ text: "Bangladesh", category: "B" }, { text: "Belgium", category: "B" },
{ text: "Canada", category: "C" }, { text: "China", category: "C" },
{ text: "Denmark", category: "D" }, { text: "Dominica", category: "D" },
{ text: "Europe", category: "E" }, { text: "Egypt", category: "E" },
{ text: "India", category: "I" }, { text: "Indonesia", category: "I" },
{ text: "France", category: "F" }, { text: "Finland", category: "F" },
{ text: "Germany", category: "G" }, { text: "Greece", category: "G" },
{ text: "Japan", category: "J" }, { text: "Jordan", category: "J" },
{ text: "Madagascar", category: "M" }, { text: "Midway Islands", category:
"M" },
{ text: "Nepal", category: "N" }, { text: "Netherlands", category: "N" },
{ text: "Qatar", category: "Q" }, { text: "Romania", category: "R" },
{ text: "Scotland", category: "S" }, { text: "Tibet", category: "T" },
{ text: "Zambia", category: "Z" }, { text: "Zimbabwe", category: "Z" }
];
$(function () {
var autocompleteInstance =new ej.Autocomplete($("#autocomplete"), {
dataSource: countries,
filterType: ej.filterType.Contains,
fields: { text: "text", groupBy: "category" }
});
});
}
```

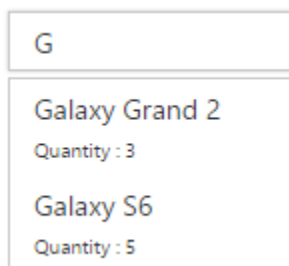


## Templates

The suggestion list can be customized based on different needs using templates. The desired templates can be defined using the “template” property.

### HTML

```
<input type="text" id="autocomplete" />
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AutocompleteComponent{
var mobileList = [
{ pName: "Galaxy Grand 2", quantity: "3" },
{ pName: "Galaxy S6", quantity: "5" },
{ pName: "IPhone S6", quantity: "8" },
{ pName: "Ipod Mini", quantity: "3" }, ];
$(function () {
var autocompleteInstance =new ej.Autocomplete($("#autocomplete"), {
dataSource: mobileList,
fields: { text: "pName" },
template: "<div><div class='product-text'>${pName}</div> <span
class='product-quantity' style='font-size:10px'> Quantity :
${quantity}</span></div>"
});
});
}
```



## Validation

You can validate the Autocomplete value on form submission by applying “validationRules” and “validationMessage” to the Autocomplete.

---

**Note:** [jquery.validate.min](#) script file should be referred for validation, for more details, refer [here](#).

---

#### Validation Rules

The validation rules help you to verify the selected text by adding validation attributes to the input element. This can be set by using [validationRules](#) property.

#### Validation Messages

You can set your own custom error message by using [validationMessage](#) property. To display the error message, specify the corresponding annotation attribute followed by the message to display.

---

**Note:** jQuery predefined error messages to that annotation attribute will be shown when this property is not defined. The below given example explain this behavior of 'required' attribute,

---

When you initialize the Autocomplete widget, it creates an input hidden element which is used to store the selected items value. Hence, the validation is performed based on the value stored in this hidden element.

Required field and min value validation is demonstrated in the below given example.

#### HTML

```
<form id="form1">
<input type="text" id="autocomplete1" />
<input type="submit" value="Validate" />
</form>
```

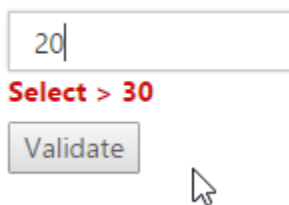
#### JAVASCRIPT

```
$.validator.setDefaults({
ignore: [],
errorClass: 'e-validation-error', // to get the error message on jQuery
validation
errorPlacement: function (error, element) {
$(error).insertAfter(element.closest(".e-widget"));
}
// any other default options and/or rules
});
//If necessary, we can create custom rules as below. here method defined for
min
$.validator.addMethod("min",
function (value, element, params) {
if (!/Invalid|NaN/.test(value)) {
return parseInt(value) > params;
}
}, 'Must be greater than 30. ');
$(function() {
var items = [{
text: "10",
value: 10
}, {
text: "20",
value: 20
}, {
text: "30",
value: 30
}, {
text: "40",
```

```

value: 40
}, {
text: "50",
value: 50
}];
var autocompleteInstance = new ej.Autocomplete($("#autocomplete1"), {
dataSource: items,
fields: {
text: "text"
},
validationRules: {
required: true,
min: 30
},
validationMessage: {
required: "* Required",
min: "Select > 30"
}
});
});

```



## Accessibility

### Keyboard Interaction

Autocomplete widget is completely compatible to work with Keyboard as an alternative for mouse. The keyboard interaction will be enabled by default in the Autocomplete widget. Press Alt+J to focus the Autocomplete element.

Please refer the below table for details about short cut keys and its corresponding usage.

| Shortcut Key | Usage  |
|--------------|--|
| Down         | Moves to next item in the suggestion list  |
| Up           | Moves to previous item in the suggestion list  |
| Enter        | Selects the focused item   |
| Ctrl + Down  | Opens the suggestion list  |
| ESC          | Closes the suggestion list   |
| Page Down    | Moves to the item in next page (i.e.,) scrolls down to next set of items when the popup is opened/closed |

|   |  |
|---|--|
| Page Up   | Moves to the item in previous page (i.e.,) scrolls up to previous set of items when the popup is opened/closed |
| Left Arrow, Right Arrow, Home, End, Number lock | No actions   |
| Tab   | Close the popup if it is opened and focuses to next DOM element  |
| Shift + Tab                                     | Focuses the previous DOM element   |

## Barcode

### Overview

The Syncfusion Essential JS Barcode widget enables rendering of one dimension and two dimension barcodes in web page. Barcode provides you a simple and inexpensive method of encoding text information that can be easily read by electronic readers.

### Key Features

- Supports 10 one-dimensional barcodes including Code 39 and Code 32 barcodes.
- Supports 2 two-dimensional barcodes such as QR and Data Matrix barcodes.

### Getting Started

**Essential JavaScript Barcode** widget provides support to create Barcode within your web page. This section explains briefly about how to create an **Barcode** in your application with **Typescript**.

#### Create Barcode in Typescript

Create an HTML page and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

Create a **Barcode** element within the body of the HTML document where the widget needs to be rendered.

#### HTML



```
<div id="targetElement">  
  <div id="Barcode"></div>  
</div>
```

Initialize the Barcode in ts file by using the `ej.datavisualization.Barcode` method.

### HTML

```
/// <reference path="../../tsfiles/jquery.d.ts" />  
/// <reference path="../../tsfiles/ej.web.all.d.ts" />  
module Barcodecomponent {  
  $(function () {  
    var barcodesample = new ej.datavisualization.Barcode($("#Barcode"), {  
      text:"http://www.syncfusion.com"  
    });  
  });  
}
```

The following screenshot illustrates the output of the above code example.



## BulletGraph

### Overview

**BulletGraphs** are easy to interpret and also it conveys much more information to the user by visualizing the data in a small amount of space. It is generally used to compare a primary measure to one or more other measures in the context of qualitative ranges of performance. Occasionally, the **BulletGraphs** are also used to compare the same measure across multiple categories.

### Key Features

- **Data sources:** Bind the **BulletGraph** control with an array of JSON objects or `ej.DataManager`.
- **Tooltip with template:** Template for tooltip support has been included to customize the tooltip.
- **FlowDirection:** The control can be rendered either in a “forward” or “backward” direction.
- **Orientation:** The control can be oriented either in “horizontal” or “Vertical” mode.
- **Binding Range Stroke to ticks and labels:** The Range colors can be bound to its underlying ticks and labels when the properties “`applyRangeStrokeToLabels`” and “`applyRangeStrokeToTicks`” are enabled.
- **Multiple Measures:** Provided support to render multiple feature measure bars as well as multiple comparative measure symbols.
- **Animation:** Animation support provided for the feature measure bars and the comparative measure symbols.

## Getting Started

For common getting started of TypeScript , you can refer [here](#).

The default type definition file ej.web.all.d.ts needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the jquery.d.ts file in your TypeScript application, which can be downloaded from [here](#).

## Adding Script Reference

Create an **HTML** page and add the scripts references in the order mentioned in the following code example.

### HTML

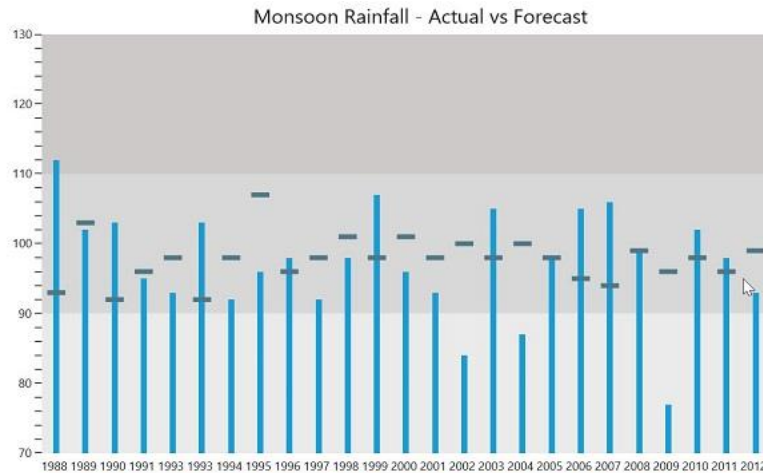
```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion }}
/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

In the above code, `ej.web.all.min.js` script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

## Create your first BulletGraph in TypeScript

This section encompasses the details on how to configure the BulletGraph control in your application. It also allows you to learn how to pass the required data to it and customize its various options according to your requirements.

In the following screenshot, a BulletGraph is used to compare the actual monsoon rainfall received in a state versus its forecasted values for the years ranging from 1988 to 2013.



1. Create a

tag.

### HTML

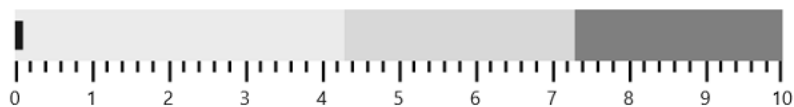
```
<html> <body> <div id="BulletGraph"></div> </body> </html>
```

2. Initialize the BulletGraph in ts file by using the `ej.BulletGraph` method.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module BulletgraphComponent {
  $(function () {
    var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"));
  });
}
```

Execute the above code to display the BulletGraph. To customize the measure bars in the BulletGraph, you can pass the data either locally or remotely.



### Provide Required Data

You can customize the values of feature and comparative measure bars in a BulletGraph, either locally or remotely. The category data is optional, and it is used to display label values parallel to the measure bars.

Assign the data in LocalData variable to the DataSource property of BulletGraph as illustrated in the following code example.

### JAVASCRIPT

```
var localData = [
{
```

```
value: 90, comparativeMeasureValue: 100,  
category: 2013  
,  
{  
value: 93, comparativeMeasureValue: 99,  
category: 2012  
,  
{  
value: 98, comparativeMeasureValue: 96,  
category: 2011  
,  
{  
value: 102, comparativeMeasureValue: 98,  
category: 2010  
,  
{  
value: 77, comparativeMeasureValue: 96,  
category: 2009  
,  
{  
value: 99, comparativeMeasureValue: 99,  
category: 2008  
,  
{  
value: 106, comparativeMeasureValue: 94,  
category: 2007  
,  
{  
value: 105, comparativeMeasureValue: 95,  
category: 2006  
,  
{  
value: 98, comparativeMeasureValue: 98,  
category: 2005  
,  
{  
value: 87, comparativeMeasureValue: 100,  
category: 2004  
,  
{  
value: 105, comparativeMeasureValue: 98,  
category: 2003  
,  
{  
value: 84, comparativeMeasureValue: 101,  
category: 2002  
,  
{  
value: 93, comparativeMeasureValue: 98,  
category: 2001  
,  
{  
value: 90, comparativeMeasureValue: 96,  
category: 2000  
,  
{  
value: 95, comparativeMeasureValue: 107,
```

```
category: 1999
},
{
value: 104, comparativeMeasureValue: 98,
category: 1998
},
{
value: 102, comparativeMeasureValue: 92,
category: 1997
},
{
value: 103, comparativeMeasureValue: 98,
category: 1996
},
{
value: 100, comparativeMeasureValue: 96,
category: 1995
},
{
value: 110, comparativeMeasureValue: 92,
category: 1994
},
{
value: 100, comparativeMeasureValue: 103,
category: 1993
},
{
value: 94, comparativeMeasureValue: 93,
category: 1992
},
{
value: 91, comparativeMeasureValue: 95,
category: 1991
},
{
value: 107, comparativeMeasureValue: 103,
category: 1990
},
{
value: 101, comparativeMeasureValue: 102,
category: 1989
},
{
value: 119, comparativeMeasureValue: 112,
category: 1988
}]];
```

Once the DataSource property is assigned with the required values, you can bind the variable names used in the JSON data to the corresponding fields of the BulletGraph as shown in the following code example.

#### JAVASCRIPT

```
$(function(){
var bulletSample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
fields: {
```

```

dataSource: localData, category: "category",
featureMeasures: "value",
comparativeMeasure: "comparativeMeasureValue"
}
});
});

```

### Set Default and Scale Values

You can plot any number of measure bars within the BulletGraph by increasing the height and width of the control to locate all the measure bars within the graph. Set the `QualitativeRangeSize` and `QuantitativeScaleLength` properties according to the following code example.

By default, the BulletGraph is rendered in the horizontal orientation with its flow direction set to Forward. In this example, to achieve the desired output, change the horizontal orientation to vertical orientation with the flow direction set to Backward.

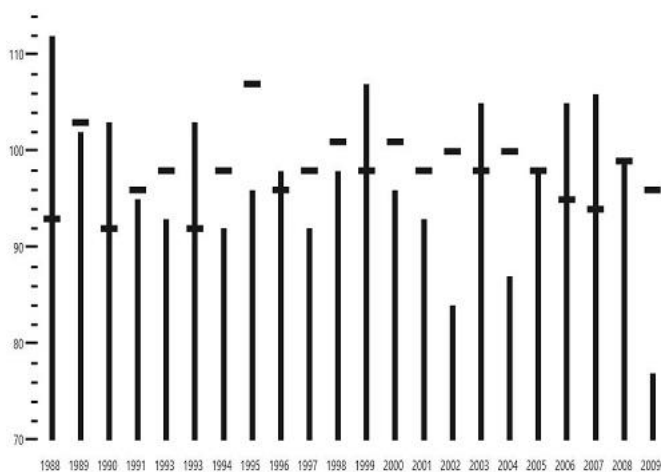
Minimum, Maximum and Interval values for the QuantitativeScale of the BulletGraph are set, as illustrated in the following code example. The ticks and labels within the scale are also positioned.

### JAVASCRIPT

```

$(function() {
var bulletSample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
quantitativeScaleSettings: {
minimum: 70,
maximum: 130,
interval: 10,
tickPosition: ej.datavisualization.BulletGraph.TickPosition.Above,
labelSettings: {
position: ej.datavisualization.BulletGraph.LabelPosition.Above
}
}
});
});

```



The above image illustrates the BulletGraph without any ranges displayed in the background.

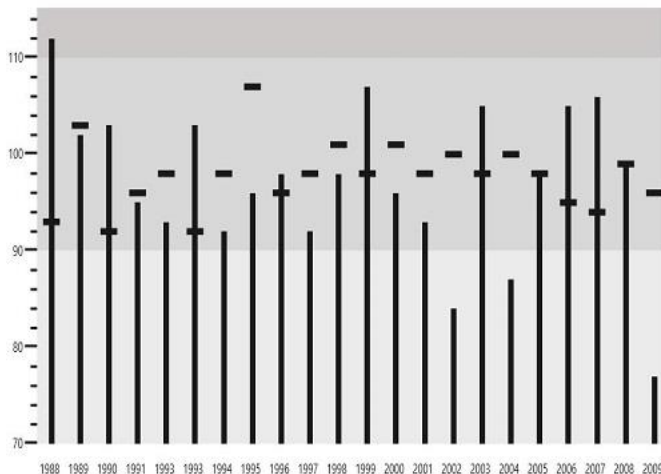
### Add Qualitative Ranges

By default, 3 ranges are displayed in the BulletGraph control during the initial rendering of the control with its default values. To customize it, you can set appropriate values for the RangeEnd and RangeStroke properties. Any number of QualitativeRanges can be added to the control.

#### JS

```
$(function() {
  var bulletSample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    qualitativeRanges: [{
      rangeEnd: 90
    },
    {
      rangeEnd: 110
    },
    {
      rangeEnd: 130, rangeStroke: "#CDC9C9"
    }
  ]
});
```

After adding QualitativeRanges to the BulletGraph, the control appears as follows.



### Ticks and Measure Bars Customization

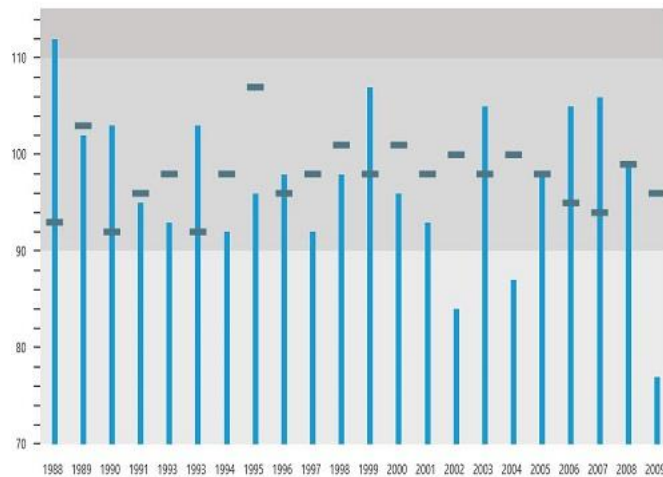
You can do the following code changes in the quantitative scale to customize the tick size, the color of the feature bar and the comparative measure symbols.

#### JS

```
$(function() {
  var bulletSample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    quantitativeScaleSettings: {
      minimum: 70,
      maximum: 130,
      interval: 10,
      tickPosition: ej.datavisualization.BulletGraph.TickPosition.Above,
      labelSettings: {
        position: ej.datavisualization.BulletGraph.LabelPosition.Above
      }
    },
  });
```

```
majorTickSettings:{width:1, size:7},
minorTickSettings:{width:1},
comparativeMeasureSettings:{stroke:"#507786"},
featuredMeasureSettings:{stroke: "#169DD8"},
}
});
});
```

When you customize the ticks and measure bar, the BulletGraph appears as follows.



#### Add Caption and Subtitle

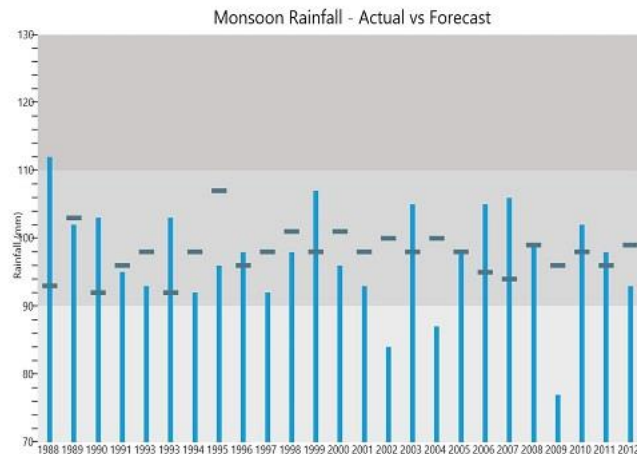
You can add the following code example to display an appropriate Caption and Subtitle to the BulletGraph.

#### JS

```
$(function(){
var bulletSample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
captionSettings: {
textAngle: 0,
location: { x: 470, y: 270 },
text: "Monsoon Rainfall - Actual vs Forecast",
font: { fontFamily: 'Segoe UI', size: '20px',
fontWeight: 'regular', opacity: 1 },
subTitle: {
textAngle: -90,
text: "Rainfall (mm)", location: { x: 180, y: 4 },
font: { fontFamily: 'Segoe UI', size: '14px',
fontWeight: 'regular', opacity: 1}
}
}
});
});
```

The following screenshot displays a BulletGraph with a Caption and Subtitle.





### Show Tooltip

You can use a Tooltip in your application to display the values of forecasted rainfall, actual rainfall received in millimeter and also the appropriate year. The Tooltip Visible property is set to true to enable the Tooltip option. To set the template Tooltip, you can pass the template id to it as illustrated in the following code example.

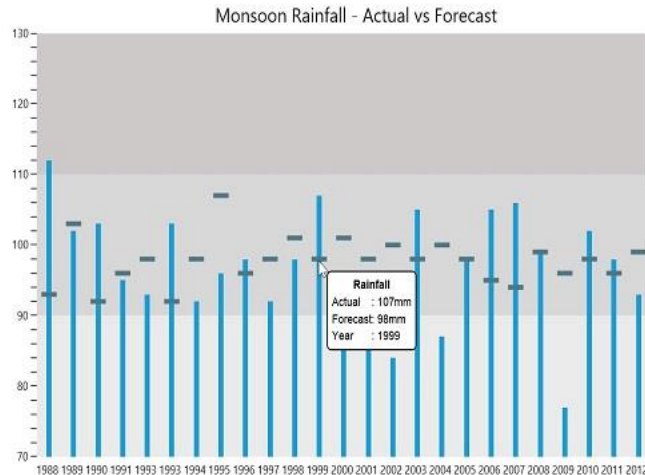
### JAVASCRIPT

```
$(function() {
    var bulletSample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
        tooltipSettings: { visible: true, template: "Tooltip" },
    });
});
```

### HTML

```
<div id="Tooltip" style="display:none; width:125px;padding-top: 10px;padding-bottom:10px">
  <div align="center" style="font-weight:bold">
    Rainfall </div>
    <table>
      <tr><td>Actual</td>
      <td> {{:currentValue}}mm</td></tr>
      <tr><td>Forecast</td>
      <td> {{:targetValue}}mm</td></tr>
      <tr><td>Year</td>
      <td> {{:category}}</td></tr>
    </table>
  </div>
```

The following screenshot displays a customized BulletGraph.



## Bullet Graph Dimensions

This section explains you on how to change the dimensions of the **Bullet Graph**. You can change various dimensions and properties of **Bullet Graph** like width, height, quantitative scale length, qualitative range size etc. By default, **Bullet Graph** uses 595 pixel width and 90 pixel height. You can customize width and height of a **Bullet Graph** using **width** and **height** properties of Bullet Graph respectively.

### Size

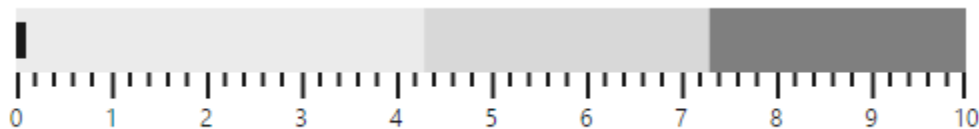
#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module BulletgraphComponent {
  $(function () {
    var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
      width: 500, height: 100
    });
  });
}

```

In the above code example, width is set as 500 pixel and height is set as 100 pixel. The output of the above code example with dimension 500 \* 100 is as follows.



### Value for performance bar

The feature measure bar value is customized using the **value** property. Default value of this property is 0.

#### JAVASCRIPT

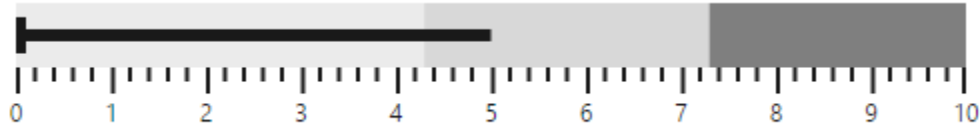
```

$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    value: 5
  });
}

```

```
});  
});
```

The following screenshot displays **Bullet Graph** with a performance measure value of 5.



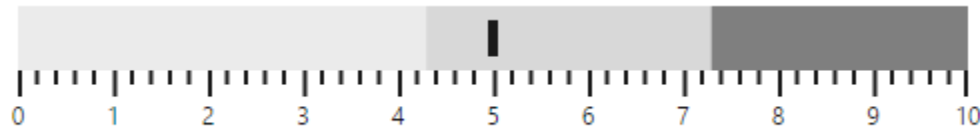
#### Comparative measure value

The **Comparative measure value** is set using **comparative measure value** property. The default value of this property is 0.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {  
    comparativeMeasureValue: 5  
  });  
});
```

The following screenshot displays **Bullet Graph** with comparative measure value of 5.



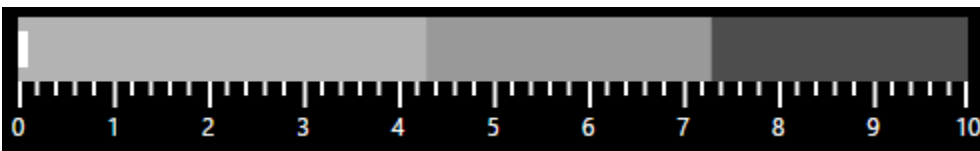
#### Theme

**Bullet Graph Theme** is customized using **theme** property. Default value is **flat light**. **Bullet Graph** supports **flat light** and **flat dark** themes. **Flat dark** theme improves **Bullet Graph** appearance when background of **Bullet Graph** container uses dark color like black.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {  
    theme: 'dark'  
  });  
});
```

The following screenshot displays **Bullet Graph** with **dark** theme



### Orientation

Bullet Graph is oriented either horizontally or vertically using **orientation** property. Default value of this property is horizontal.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {  
    orientation: 'vertical',  
    width: 100,  
    height: 550,  
    flowDirection: 'backward'  
  });  
});
```

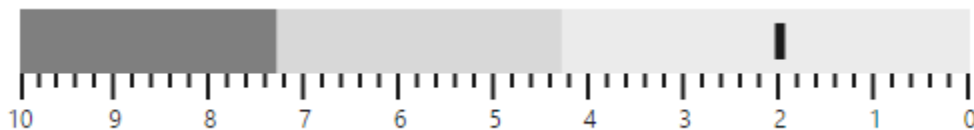
### Flow direction

The Flow direction of Bullet Graph is customized using **flowDirection** property. Default value of this property is forward. Setting forward renders Bullet Graph left to right and backward renders from right to left.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {  
    flowDirection: 'backward',  
    comparativeMeasureValue: 2  
  });  
});
```

The following screenshot displays **Bullet Graph** in a backward direction.



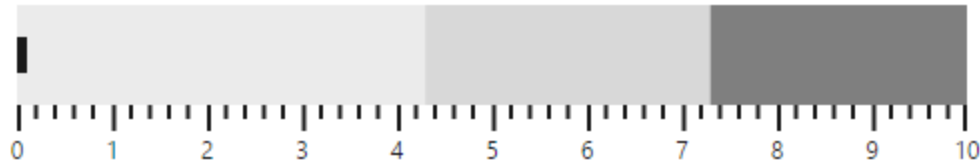
### Qualitative range size

Size of the Qualitative range is customized using **qualitative range size** property. Default value of this property is 32.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {  
    qualitativeRangeSize: 50  
  });  
});
```

The following screenshot displays **Bullet Graph** with Qualitative range of size 50



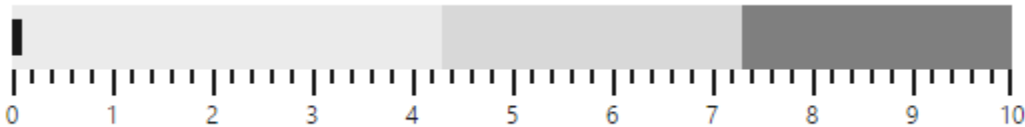
### Quantitative scale length

Length of the **QuantitativeScale** is customized using **quantitative scale length** property. Default value of this property is 475.

### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    quantitativeScaleLength: 500
  });
});
```

The following screenshot displays **Bullet Graph** with Quantitative scale length of 500.



### Bullet Graph Caption

**Bullet Graph** supports title and **subtitle** to convey what is represented in **Bullet Graph**. They are customized using **caption settings** property.

#### Title

**Title** is set to **Bullet Graph** using **text** property in **caption settings**. Caption settings also include properties like **location**, **font**, **text alignment**, **text anchor**, **text position**, **text angle** and **padding** for customizing the caption of **Bullet Graph**.

#### Location

Using **location** option, you can set the **X** and **Y** position of caption text.

#### Font

Using **font** property, you can customize **font color**, **font family**, **font style**, **font weight**, **opacity**, **size** options.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module BulletgraphComponent {
  $(function () {
    var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
      quantitativeScaleSettings: {
        location: { x: 100, y: 200 },
        minimum: 0,

```

```

maximum: 5,
interval: 0.5,
featureMeasures: [{ value: 4, comparativeMeasureValue: 3.5, category: "" }]
},
height: 700, width: 600,
captionSettings: {
  text: "Revenue YTD",
  textAngle: 0,
  location: {
    x: 10, y: 220
  },
  font: {
    color: 'gray',
    fontFamily: 'Segoe UI',
    fontStyle: 'bold',
    size: '14px',
    fontWeight: 'regular',
    opacity: 1
  }
}
});
});
}

```

The following screenshot displays a **Bullet Graph** with customized caption using the above code.



### Subtitle

**Subtitle** is added to **Bullet Graph** using **text** property of **subtitle** in **captionSettings**. **Subtitle** also provides properties like **location**, **font**, **text alignment**, **text anchor**, **text angle** and **padding** to customize subtitle similar to caption.

### Location

Using **location** option, you can set the **X** and **Y** position of the subtitle text in the caption settings.

### Font

Using **font** property, you can customize **font color**, **font family**, **font style**, **font weight**, **opacity**, **size** options of the subtitle text.

### JAVASCRIPT

```

$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    quantitativeScaleSettings: {
      location: { x: 100, y: 200 },
      minimum: 0,
      maximum: 5,
      interval: 0.5,
      featureMeasures: [{ value: 4, comparativeMeasureValue: 3.5, category: "" }]
    },
  });
});

```

```

height: 700, width: 600,
captionSettings: {
  text: "Revenue YTD",
  textAngle: 0,
  location: {
    x: 10, y: 220
  },
  font: {
    color: 'gray',
    fontFamily: 'Segoe UI',
    fontStyle: 'bold',
    size: '14px',
    fontWeight: 'regular',
    opacity: 1
  }
});
});

```

The following screenshot displays **Bullet Graph** with a subtitle



### Indicator

You can add **Indicator** to bullet graph by enabling **visible** and setting **text** properties of **indicator** in **captionSettings**. Indicator is used to represent whether target is achieved or not with text and symbol by comparing current and target values in bullet graph.

Indicator displays a **symbol** along with text which is different from caption and subtitle. Images like logos can be used in indicator instead of symbols. Indicator has properties such as **symbol**, **text**, **text spacing**, **text angle**, **text alignment**, **text anchor**, **text position**, **location**, **padding** and **font**.

### Location

Using **location** option, you can set the **X** and **Y** position of indicator text.

### Font

Using **font** property, you can customize **font color**, **font family**, **font style**, **font weight**, **opacity**, **size** font properties of the indicator text.

### Symbol

Using **symbol** settings, you can customize the following **symbol** properties.

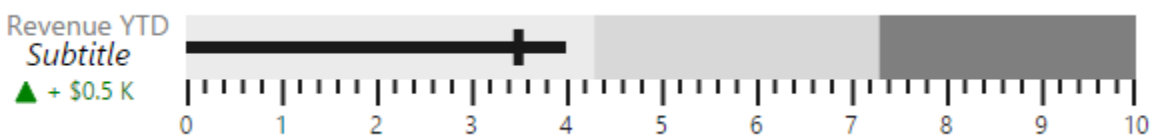
- You can customize the symbol border properties **border color** and **border width** using **border** settings.
- You can specify the color of the symbol using **color** property.
- You can set the shape of the symbol using **shape** property.
- Instead of setting shape for symbol, you can use image also. To use image as symbol, you need to set **image url** for symbol.

- You can customize the **size** of the indicator symbol using **width** and **height** properties of indicator's symbol size settings.
- To customize the opacity of the indicator symbol, you can use the **opacity** property of the symbol.

### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    captionSettings: {
      indicator: {
        visible: true,
        textAngle: 0,
        location: { x: 15, y: 240 },
        text: "+ $0.5 K",
        textSpacing: 5,
        symbol: {
          color: 'green',
          shape: 'triangle',
          imageURL: "Column.png",
          size: {
            width: 10,
            height: 10
          },
        },
        border: {
          width: 1,
          color: 'green'
        }
      },
      font: {
        color: 'green',
        fontFamily: 'Segoe UI',
        fontStyle: 'Normal ',
        size: '12px',
        fontWeight: 'regular',
        opacity: 1
      },
    }
  });
});
```

The following screenshot displays a bullet graph with indicator.



### Trim

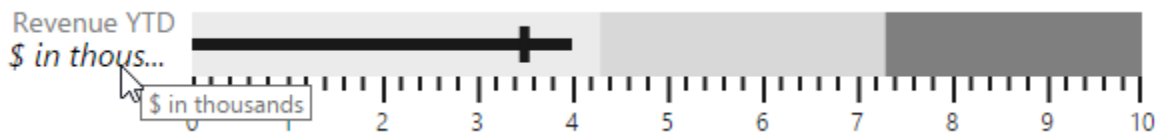
The title, subtitle and indicator text can be overlapped to the scale group. You can avoid the overlapped text by using the **enable trim** property of the captionSettings. The default value of the enableTrim is true.



**JAVASCRIPT**

```
$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    captionSettings: {
      text: 'Bullet Graph Title',
      enableTrim : true,
    }
  });
});
```

The following screenshot displays the BulletGraph with Trim.

**Text Placement**

All the caption group elements (caption, subtitle, and indicator) in the **Bullet Graph** support text positioning by using the property **textPosition** available in all caption group elements. The properties, **textAlignment** and **textAnchor** are used to customize text placement further.

**Text Position**

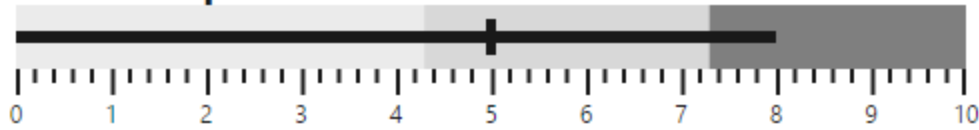
The property, **textPosition**, is used to position the text at the top, bottom, left, and right side of the quantitative scale. The default value of this property is float. By default, text can be placed at any desired location by using the **location** property.

**JAVASCRIPT**

```
$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    value: 8,
    comparativeMeasureValue: 5,
    quantitativeScaleSettings: { location: { x: 120, y: 40 }},
    height: 150,
    width: 650,
    captionSettings: {
      text: 'Bullet Graph Title',
      textPosition: 'top',
      font: {
        size: '20px',
        fontWeight: 'bold',
      }
    }
  });
});
```

The following screenshot displays the Bullet Graph with the title positioned above.

## Bullet Graph Title



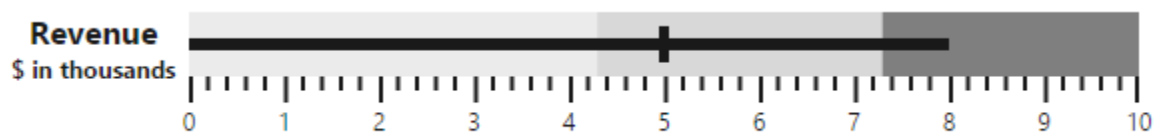
### Text Alignment

Alignment of text at different positions with respect to scale can be customized by using the **textAlignment** property. Text can be aligned in the **near**, **center**, and **far** locations of the scale. Text alignment depends upon **textPosition** property and is not applicable when the value of the **textPosition** property is **float**. The default value of the **textAlignment** property is **near**.

### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    value: 8,
    comparativeMeasureValue: 5,
    quantitativeScaleSettings: { location: { x: 120, y: 40 } },
    height: 150,
    width: 650,
    captionSettings: {
      text: 'Revenue',
      textPosition: 'left',
      textAnchor: 'middle',
      font: {
        size: '16px',
        fontWeight: 'bold',
      },
    },
    subTitle: {
      text: '$ in thousands',
      textPosition: 'left',
      textAlignment: 'center',
      font: {
        size: '12px',
        fontWeight: 'bold',
      },
    },
  });
});
```

The following screenshot displays the Bullet Graph with the title and subtitle at different alignments.

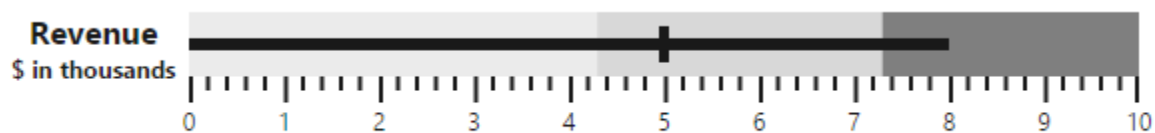


### Text Anchor

Text elements aligned at the same position are anchored by using the textAnchor property. These can be anchored at the start, middle, and end. The default value of this property is start and applicable only when two or more text elements are aligned at the same position.

### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    value: 8,
    comparativeMeasureValue: 5,
    quantitativeScaleSettings: { location: { x: 120, y: 40 }},
    height: 150,
    width: 650,
    captionSettings: {
      text: 'Revenue',
      textPosition: 'left',
      textAnchor: 'middle',
      font: {
        size: '16px',
        fontWeight: 'bold',
      },
    },
    subTitle: {
      text: '$ in thousands',
      textPosition: 'left',
      textAlignment: 'center',
      font: {
        size: '12px',
        fontWeight: 'bold',
      }
    }
  });
});
```



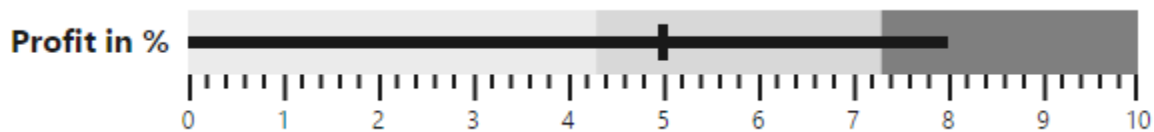
### Padding

The space required between text and quantitative scale is customized by using the padding property. The default value of this property is 5 and not applicable when the value of the textPosition property is float.

### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    value: 8,
    comparativeMeasureValue: 5,
    quantitativeScaleSettings: { location: { x: 120, y: 40 }},
    height: 150,
    width: 650,
```

```
captionSettings: {
  text: 'Profit in %',
  textPosition: 'left',
  textAlignment: 'center',
  padding: 10,
  font: {
    size: '16px',
    fontWeight: 'bold',
  }
}
});
});
```



### Data Binding

**Bullet Graph** supports binding JSON data from a remote server or data created in client-side. You can use the **fields** property to customize the data bound with **Bullet Graph**.

### Local Data

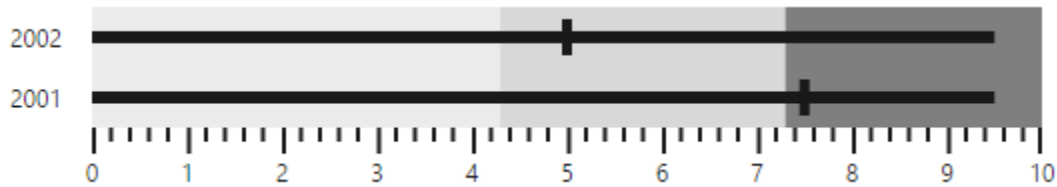
Data available in client-side (local data) can be bound with **Bullet Graph** using **fields** property. This property provides option to specify data source, fields representing progress measure bar value, comparative measure value and category value.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module BulletgraphComponent {
  var localData = [
    {
      value: 9.5, comparativeMeasureValue: 7.5,
      category: 2001
    },
    {
      value: 9.5, comparativeMeasureValue: 5,
      category: 2002
    }
  ]
  $(function () {
    var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
      qualitativeRangeSize: 60,
      quantitativeScaleSettings: {
        location: { x: 50, y: 20 },
      },
      height: 120,
      fields: {
        dataSource: localData, category: "category",
        featureMeasures: "value",
        comparativeMeasure: "comparativeMeasureValue"
      }
    });
  });
}
```

```
});
}
```

The following screenshot displays **Bullet Graph** with local data generated using TypeScript



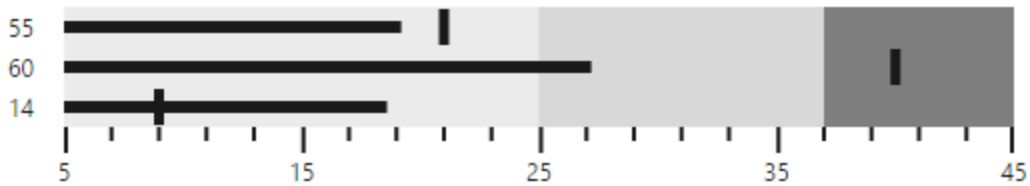
### Remote Data

**Bullet Graph** provides option to bind data from a remote server using **ejDataManager** as data source in **fields** property. A query object should also be passed to **query** property when using data manager as data source.

### JAVASCRIPT

```
//Creating data manager instance
var dataManger = new ej.DataManager({
url: "http://mvc.syncfusion.com/Services/Northwnd.svc/"
});
// Query creation
var query = ej.Query().from("Order_Details").take(3).where("UnitPrice",
ej.FilterOperators.greaterThan, 18, false)
.where("UnitPrice", ej.FilterOperators.lessThanOrEqual, 40, false)
.where("Quantity", ej.FilterOperators.greaterThan, 5, false)
.where("Quantity", ej.FilterOperators.lessThanOrEqual, 45, false);
$(function () {
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
fields: {
dataSource: dataManger,
query: query,
category: "ProductID",
featureMeasures: "UnitPrice",
comparativeMeasure: "Quantity"
},
qualitativeRanges: [{ rangeEnd: 25 }, { rangeEnd: 37 }, { rangeEnd: 45 }],
qualitativeRangeSize: 60,
quantitativeScaleSettings: {
location: { x: 50, y: 20 },
minimum: 5,
maximum: 45,
interval: 10,
},
});
});
```

The following screenshot displays a Bullet Graph bounded with data from a remote server



### Quantitative Scale

The **Quantitative Scale** appearance is customized using **quantitativeScaleSettings** property. It has properties to **customize labels, major ticks, minor ticks, comparative measure** and performance measure of the bullet graph

### Range for Quantitative Scale

**Quantitative Scale** range is set using the properties **minimum, maximum** and **interval** of **quantitativeScaleSettings** property. **Minimum** specifies the start range of the scale, **maximum** specifies the end range of scale and **interval** specifies the number of intervals between start and end range. Default values of **minimum, maximum** and **interval** are 0, 10 and 1 respectively. The number of minor ticks (ticks between intervals) are specified using **minorTicksPerInterval** property.

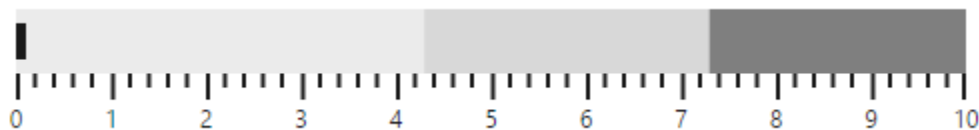
### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module BulletgraphComponent {
  $(function () {
    var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
      quantitativeScaleSettings: {
        minimum: 0,
        maximum: 10,
        interval: 1,
        minorTicksPerInterval: 4
      },
    });
  });
}

```

The following screenshot displays a **Bullet Graph** with start range 0, end range 10 and interval 1 with 4 minor ticks per interval



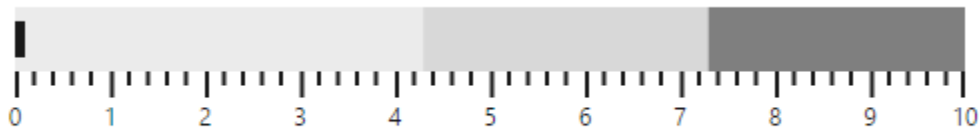
### Quantitative scale location

Bullet Graph does not position Quantitative scale automatically based on its size or space required for caption text, etc. By default Quantitative scale is positioned at 10 pixels from left and 10 pixels from top. Quantitative scale location is customized as per the requirement using the **location** property available in **quantitativeScaleSettings**. Using the this property you can set **X** and **Y** location of the quantitative scale.

### JAVASCRIPT

```
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
  quantitativeScaleSettings: {
    location: { x: 20, y: 20 }
  },
});
```

The following screenshot displays **Bullet Graph** with Quantitative scale at 20 pixels from left and 20 pixels from top



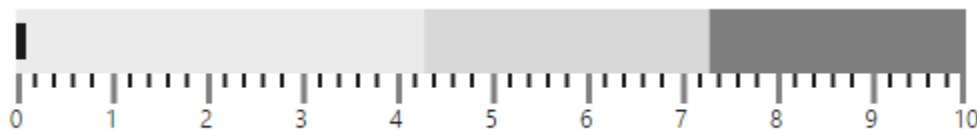
### Major ticks

Color, size and width of **Major tick** lines are customized using **major tick settings** property in **quantitativeScaleSettings**. Default value of **size** and **width** properties are 13 and 2 respectively. **Ticks** are drawn in black color by default. The property **size** represents the height of **tick** lines and **width** represents the width of **tick** lines and **ticks** color are customized using **stroke** property.

### JAVASCRIPT

```
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
  quantitativeScaleSettings: {
    majorTickSettings: {
      size: 15,
      width: 3,
      stroke: 'gray'
    }
  },
});
```

The following screenshot displays **Major ticks** in **gray** color with a width of 3 pixels and height 15 pixels.



### Minor ticks

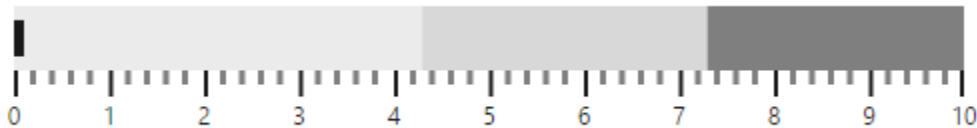
**Minor ticks** can also be customized similar to major ticks. The properties **stroke**, **width** and **size** of **minorTickSettings** are used to customize Minor ticks in quantitative scale. Stroke specifies the color of ticks, width specifies the width of ticks and size specifies the height of the ticks.

### JAVASCRIPT

```
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
  quantitativeScaleSettings: {
    minorTickSettings: {
      size: 7,
```

```
width: 3,
stroke: 'gray'
},
},
});
```

The following screenshot displays **Bullet Graph** with customized **Minor ticks** in quantitative scale



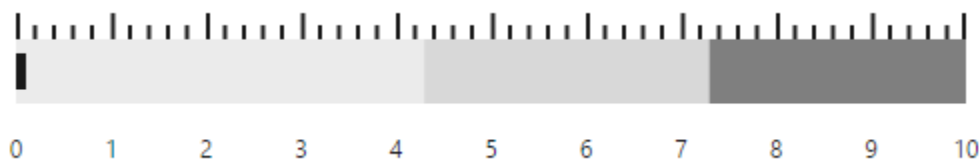
#### Tick position

**Ticks** are positioned below, above or inside the quantitative scale. By default **ticks** are positioned below the quantitative scale. The **tickPosition** property is used to customize the position of ticks in quantitative scale. **Ticks** can be placed inside the quantitative scale by setting **tickPosition** to **cross**.

#### JAVASCRIPT

```
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
  quantitativeScaleSettings: {
    tickPosition: 'above'
  },
});
```

The following screenshot displays **Bullet Graph** with ticks positioned above quantitative scale



#### Tick Placement

**QuantitativeScaleTicks** can be placed either inside or outside the scale using **tick placement** property. By default ticks are placed outside the scale.

#### JAVASCRIPT

```
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
  value: 8,
  comparativeMeasureValue: 5,
  qualitativeRangeSize: 50,
  quantitativeScaleSettings: {
    location: { x: 108, y: 10 },
    tickPlacement: 'inside',
    labelSettings: { offset: 5, size: 10, labelPrefix: '$',
    labelSuffix: ' K' },
  },
  captionSettings: {
    textAngle: 0, location: { x: 17, y: 28 }, text: "Revenue YTD",
    subTitle: {
```

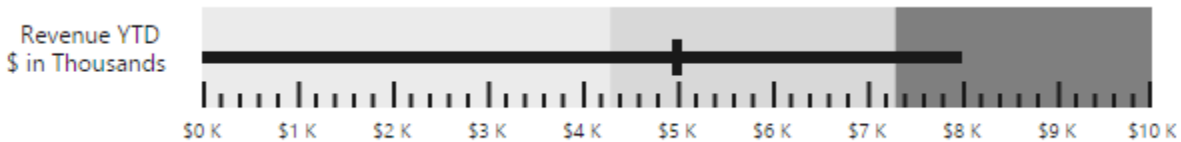


```

textAngle: 0,
text: "$ in Thousands", location: { x: 10, y: 42 }
}
}
});

```

The following screenshot displays **Bullet Graph** ticks inside **Quantitative Scale**



### Quantitative scale labels

**QuantitativeScaleLabels** are customized with prefix, suffix, font, color and size using **labelSettings** property. Following customization options are available in **labelSettings**.

- You can place the label inside or outside of the bullet graph using **label placement** property.
- Prefix and suffix to the label is added with **label prefix** and **label suffix** property.
- You can specify the horizontal or vertical padding of the labels using **offset** property.
- You can position the label either above or below the BulletGraph by using **position** property.
- To specify the size of the label text, you can use **size** property.
- You can customize the color the labels using **stroke** property.
- Using **font** option in the label settings, you can customize the **font family**, **font style**, **font weight** and **opacity** of the label text.

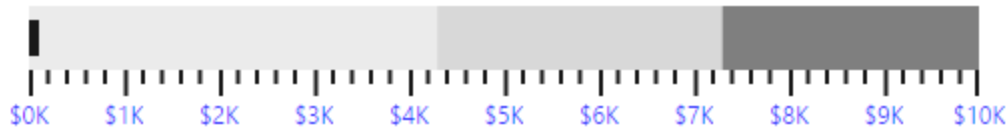
### JAVASCRIPT

```

var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
  quantitativeScaleSettings: {
    labelSettings: {
      stroke: 'blue',
      labelPrefix: '$',
      labelSuffix: 'K',
      font: {
        fontFamily: 'Segoe UI',
        fontStyle: 'bold',
        fontWeight: 'regular',
        opacity: 0.8
      },
      size: 12,
      offset: 15
    }
  },
});

```

The following screenshot displays **Bullet Graph** labels in blue color



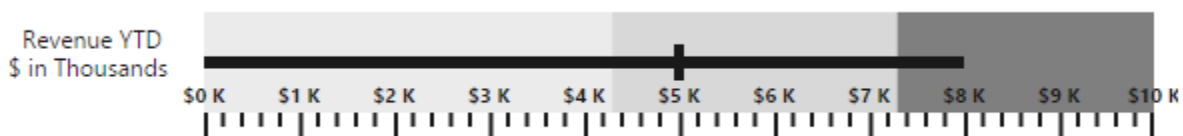
### Label Placement

**QuantitativeScaleLabels** can be placed either inside or outside the scale using **labelPlacement** property. By default labels are placed 15 pixels outside the scale.

### JAVASCRIPT

```
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
  value: 8,
  comparativeMeasureValue: 5,
  qualitativeRangeSize: 50,
  quantitativeScaleSettings: {
    location: { x: 108, y: 10 },
    labelSettings: {
      offset: 5, size: 10, labelPrefix: '$', labelSuffix: ' K', font: {
        fontWeight: 'bold' },
      labelPlacement: 'inside'
    },
  },
  captionSettings: {
    textAngle: 0, location: { x: 17, y: 28 }, text: "Revenue YTD",
    subTitle: {
      textAngle: 0,
      text: "$ in Thousands", location: { x: 10, y: 42 }
    }
  }
});
```

The following screenshot displays **Bullet Graph** labels inside **Quantitative Scale**.



### Performance measure bar

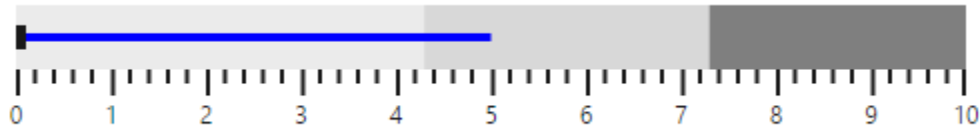
Performance measure bar is customized using **featuredMeasureSettings** in **quantitativeScaleSettings** property. Color of the bar is customized using **stroke** property and **width** using **width** property. By default bar is drawn in black color with 6 pixels of width.

### JAVASCRIPT

```
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
  value: 5,
  quantitativeScaleSettings: {
    featuredMeasureSettings: {
      stroke: 'blue',
      width: 4
    },
  },
});
```

```
},
});
```

The following screenshot displays **Bullet Graph** with customized **Performance measure bar**.



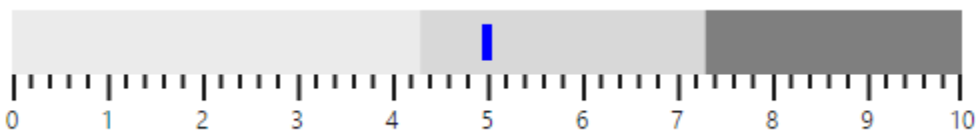
#### Comparative measure symbol

Comparative symbol color and width are customized using **comparativeMeasureSettings** through **quantitativeScaleSettings** property. Color of the symbol is customized using **stroke** property and width using **width** property. By default Comparative measure symbol is displayed in black color with a width of 5 pixels.

#### JAVASCRIPT

```
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
  comparativeMeasureValue: 5,
  quantitativeScaleSettings: {
    comparativeMeasureSettings: {
      stroke: 'blue',
      width: 5
    },
  },
});
```

The following screenshot displays **Bullet Graph** with customized **Comparative measure value**.



#### Multiple performance measures comparison

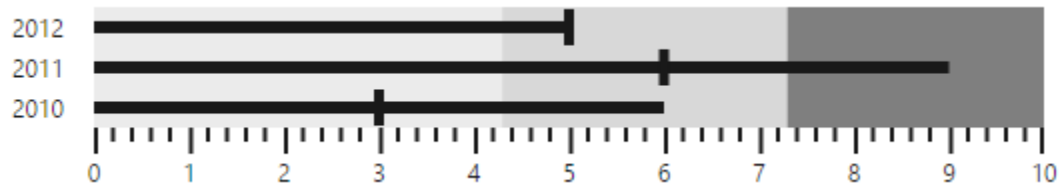
**Bullet Graph** supports comparing more than one performance at a time, given that all the comparisons are related using **featureMeasure** in **quantitativeScaleSettings** property. In feature measures, you can specify **category**, **comparative measure value** and **value** properties that are used to comparing performance.

#### JAVASCRIPT

```
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
  qualitativeRangeSize: 60,
  height: 120,
  quantitativeScaleSettings: {
    featureMeasures: [
      { value: 6, comparativeMeasureValue: 3, category: 2010 },
      { value: 9, comparativeMeasureValue: 6, category: 2011 },
      { value: 5, comparativeMeasureValue: 5, category: 2012 },
    ]
  }
});
```

```
}
});
```

The following screenshot displays **Bullet Graph** that compares 3 related performance measures.



### Qualitative Range

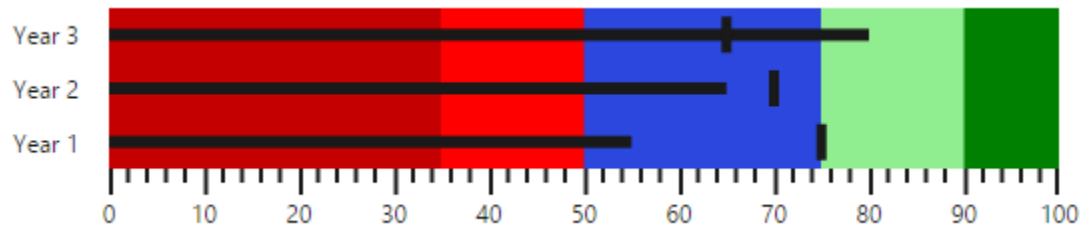
**Qualitative Range** represents the quality of a specific range in quantitative scale like good, bad and satisfactory. Color for each qualitative range is customized using **range stroke** property. The **range end** property specifies the ending point of the qualitative range. Minimum value of quantitative scale is considered as the starting point of first qualitative range and previous end points are considered as starting point for other qualitative ranges.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module BulletgraphComponent {
$(function () {
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
qualitativeRanges: [
{ rangeEnd: 35, rangeStroke: 'darkred', rangeOpacity: 0.5 },
{ rangeEnd: 50, rangeStroke: 'red', rangeOpacity: 1 },
{ rangeEnd: 75, rangeStroke: 'blue', rangeOpacity: 0.7 },
{ rangeEnd: 90, rangeStroke: 'lightgreen', rangeOpacity: 1 },
{ rangeEnd: 100, rangeStroke: 'green', rangeOpacity: 1 }
],
qualitativeRangeSize: 80,
quantitativeScaleSettings: {
location: { x: 50, y: 20 },
minimum: 0,
maximum: 100,
interval: 10,
featureMeasures: [
{ value: 55, comparativeMeasureValue: 75, category: "Year 1" },
{ value: 65, comparativeMeasureValue: 70, category: "Year 2" },
{ value: 80, comparativeMeasureValue: 65, category: "Year 3" }
]
},
height: 200
});
});
}
```

The following screenshot displays **Bullet Graph** with different qualitative ranges in different colors. In this image, range 0 to 35 represents bad performance, 35 to 50 represents average performance, 50 to

75 represents that the performance is above average, 75 to 90 represents good performance and above 90 represents excellent performance.



## User Interaction

### Animation

**Bullet Graph** supports animation that makes the performance measure bar to animate when rendering the Bullet Graph. **Animation** is enabled or disabled using **enable animation** property. By default, **Animation** is enabled in Bullet Graph.

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module BulletgraphComponent {
  $(function () {
    var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
      enableAnimation: true,
      value: 8,
      comparativeMeasureValue: 5,
    });
  });
}

```

### Responsiveness during browser resize

**Bullet Graph** is made responsive when resizing the browser by using **isResponsive** property. By default the value of this property is **true** in Bullet Graph.

### JAVASCRIPT

```

$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    isResponsive: true,
    value: 8,
    comparativeMeasureValue: 5,
  });
});

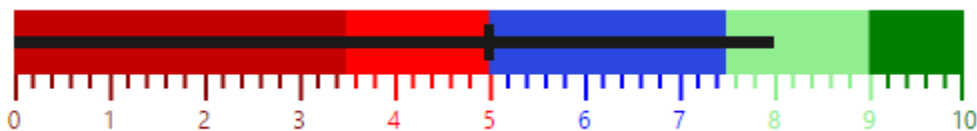
```

### Applying same color to all ticks and labels in a range

Background color for qualitative range is applied to major ticks and minor ticks of the **Bullet Graph** using **applyRangeStrokeToTicks** property. The range colors are applied to labels using **applyRangeStrokeToLabels** property. By default same colors are not applied to a qualitative range and its corresponding ticks or labels.

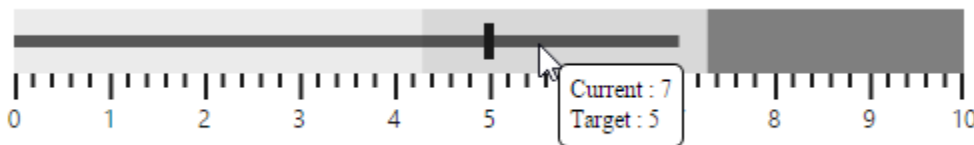
### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    applyRangeStrokeToTicks: true,
    applyRangeStrokeToLabels: true,
    qualitativeRanges: [
      { rangeEnd: 3.5, rangeStroke: 'darkred', rangeOpacity: 0.5 },
      { rangeEnd: 5.0, rangeStroke: 'red', rangeOpacity: 1 },
      { rangeEnd: 7.5, rangeStroke: 'blue', rangeOpacity: 0.7 },
      { rangeEnd: 9.0, rangeStroke: 'lightgreen', rangeOpacity: 1 },
      { rangeEnd: 10.0, rangeStroke: 'green', rangeOpacity: 1 }
    ],
    value: 8,
    comparativeMeasureValue: 5,
  });
});
```



### Tooltip

By default **Bullet Graph** displays **Tooltip** when mouse is hovered over feature measure bar. **Tooltip** is enabled or disabled using **visible** property in **tooltipSettings**.



Bullet Graph supports Tooltip template instead of default Tooltip to customize the appearance and contents of Tooltip. The Tooltip template should be a <div> element with display set to 'none', so it is displayed only when mouse is placed on feature measure bar. The id value of the <div> element should be provided as value to the **template** property in tooltipSettings of Bullet Graph to display the customized <div> element as Tooltip instead of default Tooltip. The values displayed in default Tooltip such as current value, target value and category are accessed in template <div> element by using {{currentValue}}, {{targetValue}} and {{category}} respectively.

### HTML

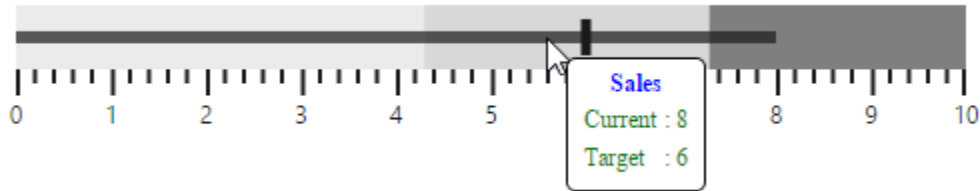
```
<div id="BulletGraphTooltip" style="display:none; width:125px; padding-top:
10px; padding-bottom:10px; color: blue">
<div align="center" style="color:blue; font-weight:bold"> Sales </div>
<table style="color:green"> <tr> <td> Current </td> <td> : </td> </tr> <tr>
<td> Target </td> <td> : </td> </tr> </table>
</div>
```

### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
```

```
value: 8,
comparativeMeasureValue: 6,
height: 150,
tooltipSettings: { template: 'BulletGraphTooltip' }
});
});
```

The following screenshot displays **Bullet Graph** with a customized **Tooltip** including a header and contents such as current value and target value in different colors.



## Methods

`destroy()`

**Destroy** method is used to destroy the bullet graph control.

Returns: void

## HTML

```
<div id="bulletgraph1">Bullet Graph</div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module BulletgraphComponent {
$(function () {
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
///...///
});
// Destroy Bullet graph
sample.destroy();
});
}
```

`redraw()`

**Redraw** method is used to redraw the bullet graph with updated values.

Returns: void

## HTML

```
<div id="bulletgraph1">Bullet Graph</div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```
module BulletgraphComponent {
  $(function () {
    var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
      //...//
    });
    // Redraws the Bullet graph
    sample.redraw();
  });
}
```

setComparativeMeasureSymbol(index, measure)

To set the comparative measure in bullet graph, you can use **setComparativeMeasureSymbol** method.

Returns: void

#### HTML

```
<div id="bulletgraph1">Bullet Graph</div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module BulletgraphComponent {
  $(function () {
    var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
      //...//
    });
    // set the value for comparative measure
    sample.setComparativeMeasureSymbol(1, 7);
  });
}
```

setFeatureMeasureBarValue(index, measure)

To set the value for feature measure bar, you can use **setFeatureMeasureBarValue** method.

Returns: void

#### HTML

```
<div id="bulletgraph1">Bullet Graph</div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module BulletgraphComponent {
  $(function () {
    var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
      //...//
    });
    // set the value for feature measure bar.
    sample.setFeatureMeasureBarValue(1, 8);
  });
}
```



## Events

### drawCaption

**drawCaption** event will fire before rendering bullet graph caption.

#### JAVASCRIPT

```
<script>
//drawCaption event for bullet graph
$(function () {
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
drawCaption: function () {
//...//
}
});
});
</script>
```

### drawCategory

**drawCategory** event will fire while rendering the category.

#### JAVASCRIPT

```
<script>
//drawCategory event for bullet graph
$(function () {
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
drawCategory: function () {
//...//
}
});
});
</script>
```

### drawComparativeMeasureSymbol

**drawComparativeMeasureSymbol** event fires on rendering the comparative measure symbol.

#### JAVASCRIPT

```
<script>
//drawComparativeMeasureSymbol event for bullet graph
$(function () {
var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
drawComparativeMeasureSymbol: function () {
//...//
}
});
});
</script>
```

### drawFeatureMeasureBar

**drawFeatureMeasureBar** event fires on rendering the feature measure bar.

#### JAVASCRIPT

```
<script>
```

```
//drawFeatureMeasureBar event for bullet graph
$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    drawFeatureMeasureBar: function () {
      //...//
    }
  });
});
</script>
```

### drawIndicator

**drawIndicator** event fires on rendering the indicator of the bullet graph.

#### JAVASCRIPT

```
<script>
//drawIndicator event for bullet graph
$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    drawIndicator: function () {
      //...//
    }
  });
});
</script>
```

### drawLabels

**drawLabels** event fires on rendering the bullet graph labels.

#### JAVASCRIPT

```
<script>
//drawLabels event for bullet graph
$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    drawLabels: function () {
      //...//
    }
  });
});
</script>
```

### drawTicks

**drawTicks** event fires while drawing the ticklines of the bullet graph.

#### JAVASCRIPT

```
<script>
//drawTicks event for bullet graph
$(function () {
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {
    drawTicks: function () {
      //...//
    }
  });
});
```

```
});  
</script>
```

### drawQualitativeRanges

**drawQualitativeRanges** event fires while rendering the qualitative ranges.

#### JAVASCRIPT

```
<script>  
//drawQualitativeRanges event for bullet graph  
$(function () {  
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {  
    drawQualitativeRanges: function () {  
      //...  
    }  
  });  
});  
</script>
```

### load

**load** event fires on loading the bullet graph.

#### JAVASCRIPT

```
<script>  
//load event for bullet graph  
$(function () {  
  var sample = new ej.datavisualization.BulletGraph($("#BulletGraph"), {  
    load: function () {  
      //...  
    }  
  });  
});  
</script>
```

## Button

### Overview

The TypeScript Button control allows you to perform an action by clicking on it. The TypeScript Button control has the feature of displaying both text and images.

### Key Features

- Trendy Look: Rich Appearance with theme support.
- RTL: Supports for right to left alignment.
- Repeat-Button: Supports the rising of click event repeatedly.
- Text and Image: Supports both text and image as Button content.
- Built-in Icon: Supports the built-in icon libraries.
- Easy Customization: The customization of Button control to any form is made simple.

### Create a simple Button in TypeScript

You can create a **TypeScript** application with the help of the given [Typescript Getting Started Documentation](#).

Create an **HTML** page and add the scripts references in the order, mentioned in the above link Create the **Button** control as follows.

#### HTML

```
<body>
<table>
<tr>
<td >My First Button</td>
<td>
<input id="button" value="button"/>
</td>
</tr>
</table>
</body>
```

#### JS

```
///<reference path="tsfiles/jquery.d.ts" />
///<reference path="tsfiles/ej.web.all.d.ts" />
module ButtonComponent {
$(function () {
var sample = new ej.Button($("#button"));
});
}
```

### Configuring Button Properties

This section encompasses the details on how you can configure the Button control in your application and customize it with various properties such as various size, resizing the **Button** according to your requirement.

To render the button with required text, size and with rounded corner add the following code example in your TS file.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ButtonComponent {
$(function () {
var sample = new ej.Button($("#button"), { text: "Button",
showRoundedCorner: true });
});
}
```

The following screenshot illustrates the **Button** control with text, size and rounded corner properties.

My First Button

BUTTON

## Easy Customization

Button is used in all applications. Button size, content and content location is varied according to each application. Here you can see some customizable option for button that can perform easily.

### Button Size

You can render the button in different sizes. Here, you have some predefined size options for rendering a button with different sizes in easiest way. Each size option has different height and width. Mainly it avoids the complexity in rendering button with complex **CSS** class.

List of predefined button size

| Button Types | Description   |
|--------------|---|
| Normal       | Creates button with content size.                                 |
| Mini         | Creates button with Built-in mini size height, width specified.   |
| Small        | Creates button with Built-in small size height, width specified.  |
| Medium       | Creates button with Built-in medium size height, width specified. |
| Large        | Creates button with Built-in large size height, width specified.  |

Apart from the above mentioned predefined size option, you can set your own width and height for button using **height** and **width** property.

The following steps explains you the details about rendering the button with different size options.

In the **HTML** page, add the following button elements to configure button widget.

### HTML

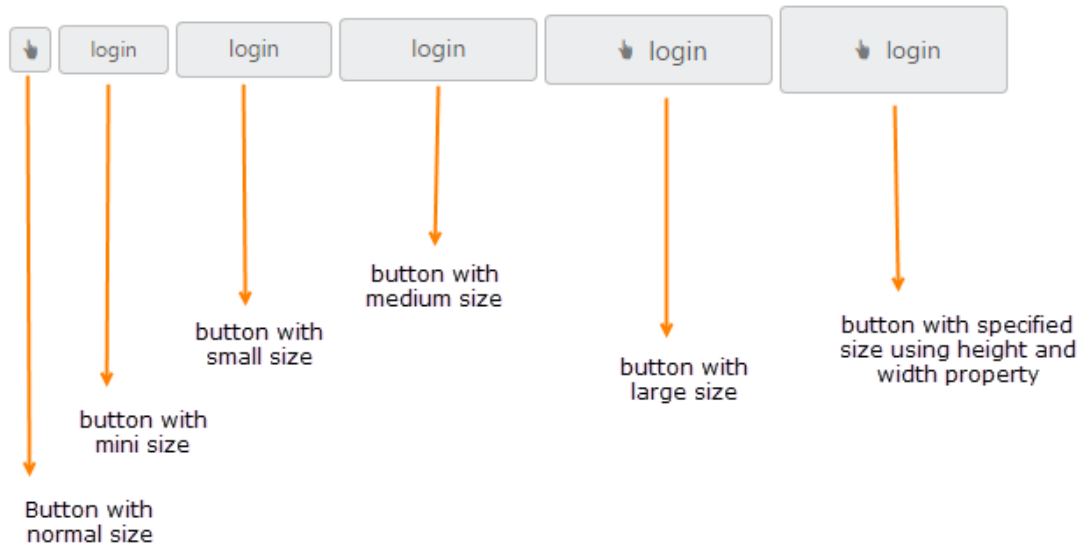
```
<button id="button_normal">login</button>
<button id="button_mini">login</button>
<button id="button_small">login</button>
<button id="button_medium">login</button>
<button id="button_large">login</button>
<button id="button_custom">login</button>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ButtonComponent {
    $(function () {
        var basicButton = new ej.Button($("#button_normal"), {
            //normal size type is used
            size: "normal",
```

```
showRoundedCorner: true,
contentType: "imageonly",
prefixIcon: "e-icon e-handup"
});
var basicButton1 = new ej.Button($("#button_mini"), {
showRoundedCorner: true,
//mini size type is used
size: "mini"
});
var basicButton2 = new ej.Button($("#button_small"), {
showRoundedCorner: true,
//small size type is used
size: "small"
});
var basicButton3 = new ej.Button($("#button_medium"), {
showRoundedCorner: true,
//medium size type is used
size: "medium"
});
var basicButton4 = new ej.Button($("#button_large"), {
//large size type is used
size: "large",
showRoundedCorner: true,
contentType: "textandimage",
prefixIcon: "e-icon e-handup"
});
var basicButton4 = new ej.Button($("#button_custom"), {
//button with user given height and width
height: 50,
width: 130,
showRoundedCorner: true,
contentType: "textandimage",
prefixIcon: "e-icon e-handup"
});
});
```

Execute the above code to render the following output.



### Content Type

The content of the **Button** is mainly text and images. Instead of using complex **CSS** classes to render **Button** with different content types, you can use some predefined content type options provided for button control. Using this content types you can easily add different types of content for button. **Button** supports the following content types.

List of content types for button

| Content Types  | Description                                       |
|----------------|---|
| TextOnly       | Supports only for text content only.              |
| ImageOnly      | Supports only for image content only              |
| ImageBoth      | Supports image for both ends of the button.       |
| TextAndImage   | Supports image with the text content.             |
| ImageTextImage | Supports image with both ends and middle in text. |

### Prefix and Suffix icons

Icons inside the **Button** is added easily using **prefixIcon** and **suffixIcon** property. Location of the icon in button is a necessary thing and you can easily customize it using the following mentioned options.

**Button** control also supports the Built-in icon libraries. The **ej.widgets.core.min.css** contains definitions for important icons that can be used in buttons. Simply you can use these Built-in icons by mentioning the icon class name as value in **prefixIcon** and **suffixIcon** property. You can use any font icons that are defined in **ej.widgets.core.min.css**. It avoids the complexity in specifying icon using sprite image and **CSS**.

For example the following mentioned Built-in **CSS** class are used to show the font icons that is used by media player.

- e-mediaback
- e-mediaforward
- e-medianext
- e-mediaprev
- e-mediaeject
- e-mediaclose
- e-mediapause
- e-mediaplay

#### Prefix Icon

It inserts the icon at the starting position of button. After this prefix icon, you can use text or suffix icon.

#### Suffix Icon

It inserts the icon at the ending position of button. Before this suffix icon, you can use text or prefix icon.

The following steps explain you the details about rendering the **Button** with above mentioned **content type**, **prefix** and **suffix icon** options

In the **HTML** page, add the following button elements to configure **Button** widget.

#### HTML

```
<button id="button_imageonly">login</button>
<button id="button_textonly">login</button>
<button id="button_imageboth">login</button>
<button id="button_textandimage">login</button>
<button id="button_imagetextimage">login</button>
<br />
<br />
<button id="button_imageonly_small">login</button>
<button id="button_textonly_small">login</button>
<button id="button_imageboth_small">login</button>
<button id="button_textandimage_small">login</button>
<button id="button_imagetextimage_small">login</button>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ButtonComponent {
$(function () {
var basicButton = new ej.Button($("#button_imageonly"), {
//only image is used as content
contentType: "imageonly",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup"
});
var basicButton1 = new ej.Button($("#button_textonly"), {
//only text is used as content
contentType: "textonly",
showRoundedCorner: true,
});
var basicButton2 = new ej.Button($("#button_imageboth"), {
//only images in both end is used as content
contentType: "imageboth",
```



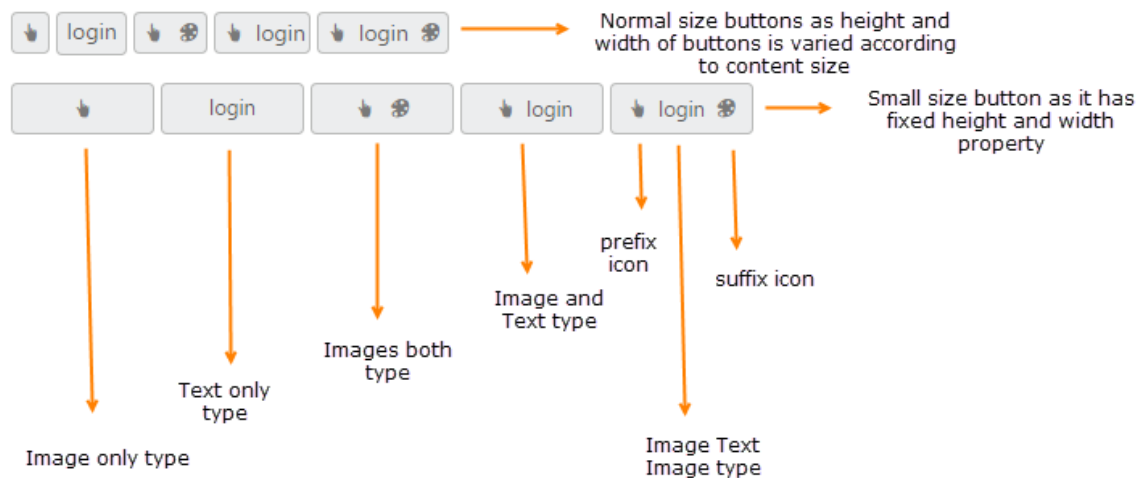
```
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
suffixIcon: "e-icon e-palette"
});
var basicButton3 = new ej.Button($("#button_textandimage"), {
//text and image is used as content
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup"
});
var basicButton4 = new ej.Button($("#button_imagetextimage"), {
//images in both end and text in center is used as content
contentType: "imagetextimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
suffixIcon: "e-icon e-palette"
});
var basicButton5 = new ej.Button($("#button_imagetextimage"), {
$("#button_imageonly_small").ejButton({
size: "small",
//only image is used as content
contentType: "imageonly",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup"
});
var basicButton6 = new ej.Button($("#button_imagetextimage"), {
$("#button_textonly_small").ejButton({
size: "small",
//only text is used as content
contentType: "textonly",
showRoundedCorner: true,
});
var basicButton7 = new ej.Button($("#button_imageboth_small"), {
size: "small",
//only images in both end is used as content
contentType: "imageboth",
showRoundedCorner: true,
//It specifies the image in prefix location
prefixIcon: "e-icon e-handup",
//It specifies the image in suffix location
suffixIcon: "e-icon e-palette"
});
var basicButton8 = new ej.Button($("#button_textandimage_small"), {
size: "small",
//text and image is used as content
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup"
});
var basicButton9 = new ej.Button($("#button_imagetextimage_small"), {
size: "small",
//images in both end and text in center is used as content
contentType: "imagetextimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
suffixIcon: "e-icon e-palette"
});
```

```

}};
}

```

Execute the above code to render the following output.



### Image Position

To provide the best look and feel for **Button**, position of button images is an important customizable option. With **imagePosition** property you can easily customize the position of images inside button without using any complex **CSS**. **imagePosition** property is applicable only with the **textandimage** of **contentType** property. This property supports the following values.

List of values supported by ImagePosition property

| ImagePosition | Description   |
|---------------|---|
| ImageLeft     | Support for aligning text in right and image in left. |
| ImageRight    | Support for aligning text in left and image in right. |
| ImageTop      | Support for aligning text in bottom and image in top. |
| ImageBottom   | Support for aligning text in top and image in bottom. |

The following steps explain you the details about rendering the **Button** with the above mentioned image Position options.

In the **HTML** page, add the following button elements to configure **Button** widget.

### HTML

```

<button id="button_imageleft_normal">login</button>
<button id="button_imageleft_mini">login</button>
<button id="button_imageleft_small">login</button>
<button id="button_imageleft_medium">login</button>
<button id="button_imageleft_large">login</button>

```

```

<br />
<br />
<button id="button_imageright_normal">login</button>
<button id="button_imageright_mini">login</button>
<button id="button_imageright_small">login</button>
<button id="button_imageright_medium">login</button>
<button id="button_imageright_large">login</button>
<br />
<br />
<button id="button_imagetop">login</button>
<button id="button_imagebottom">login</button>

```

## JAVASCRIPT

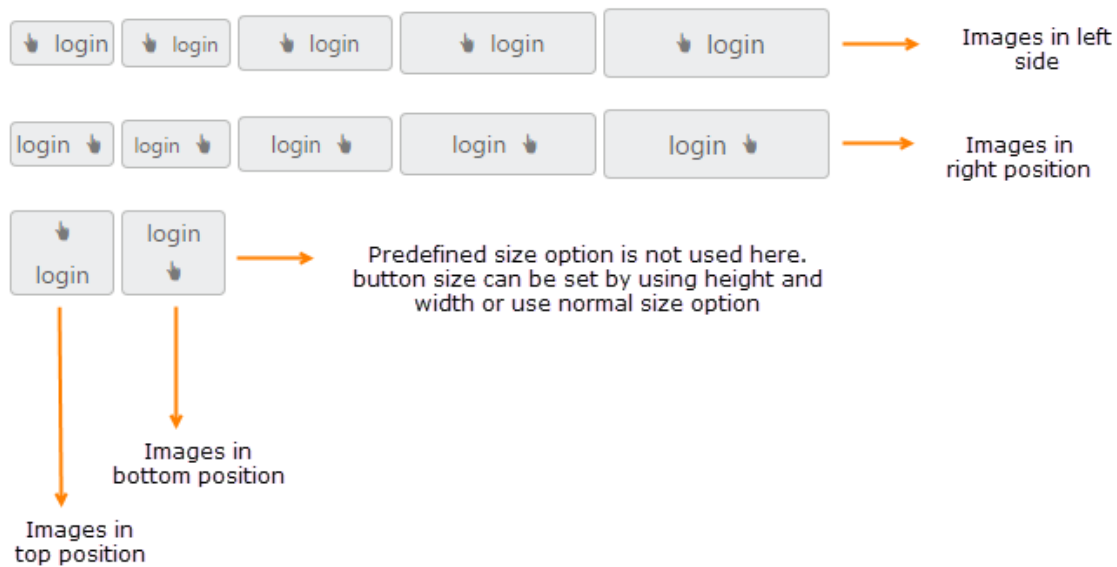
```

//Using imagePosition property you can render the button images with
different position
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ButtonComponent {
$(function () {
var basicButton = new ej.Button($("#button_imageleft_normal"), {
size: "normal",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
});
var basicButton1 = new ej.Button($("#button_imageleft_mini"), {
size: "mini",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
});
var basicButton2 = new ej.Button($("#button_imageleft_small"), {
size: "small",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
});
var basicButton3 = new ej.Button($("#button_imageleft_medium"), {
size: "medium",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
});
var basicButton4 = new ej.Button($("#button_imageleft_large"), {
size: "large",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
});
var basicButton5 = new ej.Button($("#button_imageright_normal"), {

```

```
size: "normal",
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
});
var basicButton6 = new ej.Button($("#button_imageright_mini"), {
size: "mini",
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
});
var basicButton7 = new ej.Button($("#button_imageright_small"), {
size: "small",
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
});
var basicButton8 = new ej.Button($("#button_imageright_medium"), {
size: "medium",
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
});
var basicButton9 = new ej.Button($("#button_imageright_large"), {
size: "large",
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
});
var basicButton10 = new ej.Button($("#button_imagetop"), {
imagePosition: "imagetop",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
width: 60
});
var basicButton11 = new ej.Button($("#button_imagebottom"), {
imagePosition: "imagebottom",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
width: 60
});
});
});
}
```

Execute the above code to render the following output.



### Theme support

You can control the style and appearance of **Button** control based on **CSS** classes. In order to apply styles to the **Button** control, you can refer two files namely, **ej.widgets.core.min.css** and **ej.theme.min.css**.

When you refer the **ej.widgets.all.min.css** file, then it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.widgets.all.min.css** is the combination of these two.

By default, there are 16 themes support available for **Button** control.

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap
- high-contrast-01
- high-contrast-02
- material
- office-365

### Custom CSS

You can customize the appearance of **Button** control using **CSS** class. Define a **CSS** class as per requirement and assign the class name to **cssClass** property.

The following steps explain you the details about rendering the **Button** with custom **CSS**.

In the **HTML** page, add the following button elements to configure **Button** widget.

### HTML

```
<button id="button_customCss1">login</button>
<button id="button_customCss2">login</button>
<button id="button_customCss3">login</button>
<button id="button_customCss4">login</button>
<button id="button_customCss5">login</button>
```

### JAVASCRIPT

```
//implement custom CSS for each button
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ButtonComponent {
$(function () {
var basicButton = new ej.Button($("#button_customCss1"), {
cssClass: "customCss1",
size: "small",
showRoundedCorner: true,
contentType: "textandimage",
prefixIcon: "e-icon e-handup",
});
var basicButton = new ej.Button($("#button_customCss2"), {
cssClass: "customCss2",
size: "small",
showRoundedCorner: true,
contentType: "textandimage",
prefixIcon: "e-icon e-handup"
});
var basicButton = new ej.Button($("#button_customCss3"), {
cssClass: "customCss3",
size: "small",
showRoundedCorner: true,
contentType: "textandimage",
prefixIcon: "e-icon e-handup"
});
var basicButton = new ej.Button($("#button_customCss4"), {
cssClass: "customCss4",
size: "small",
showRoundedCorner: true,
contentType: "textandimage",
prefixIcon: "e-icon e-handup"
});
var basicButton = new ej.Button($("#button_customCss5"), {
cssClass: "customCss5",
size: "small",
showRoundedCorner: true,
contentType: "textandimage",
prefixIcon: "e-icon e-handup"
});
});
}
```

Configure the **CSS** styles to apply on buttons.

### CSS

```
<style type="text/css" class="cssStyles">
/* Customize the button background */
.e-button.customCss1 {
background-color: #121111;
}
.e-button.customCss2 {
background-color: #94bbd5;
}
.e-button.customCss3 {
background-color: #f3533c;
}
.e-button.customCss4 {
background-color: #d1eed;
}
.e-button.customCss5 {
background-color: #deb66e;
}
/* Customize the button image & text color */
.e-button.customCss1.e-btn.e-select .e-icon, .e-button.customCss1.e-btn.e-
select .e-btntxt {
color: #94bbd5;
}
.e-button.customCss2.e-btn.e-select .e-icon, .e-button.customCss2.e-btn.e-
select .e-btntxt {
color: #121111;
}
.e-button.customCss3.e-btn.e-select .e-icon, .e-button.customCss3.e-btn.e-
select .e-btntxt {
color: #cef6f7;
}
.e-button.customCss5.e-btn.e-select .e-icon, .e-button.customCss5.e-btn.e-
select .e-btntxt {
color: #534f4f;
}
</style>
```

Execute the above code to render the following output.



## Button Type

**Button** is used as normal click able button, submitting form data, resetting the form data to its initial value. According to the usage of button, you can render the button in three types. Using the **type** property, you can easily render the button in following types.

List of Button types

| Button Types | Description   |
|--------------|---|
| button       | The button is a click able button   |
| submit       | The button is a submit button (submits form-data)                         |
| reset        | The button is a reset button (resets the form-data to its initial values) |

The following steps explains you the details about rendering the Button with above mentioned button types.

In the HTML page, add the following button elements to configure Button widget.

### HTML

```
<button id="buttonType_button">button</button>
<br />
<br />
<button id="buttonType_submit">submit</button>
<br />
<br />
<button id="buttonType_reset">reset</button>
```

### JAVASCRIPT

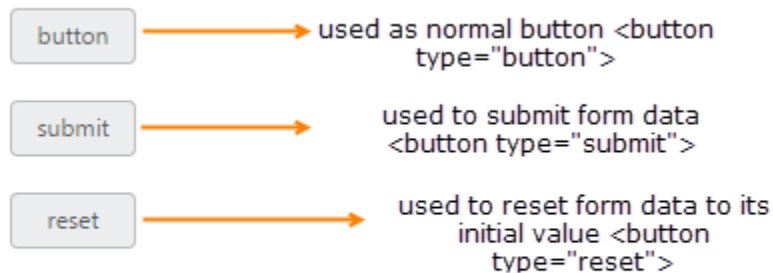
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ButtonComponent {
$(function () {
var basicButton = new ej.Button($("#buttonType_button"), {
size: "mini",
//this type specifies the normal click able button
type: "button",
showRoundedCorner: true
});
var basicButton1 = new ej.Button($("#buttonType_submit"), {
size: "mini",
//this button is used to submit form data
type: "submit",
showRoundedCorner: true
});
var basicButton2 = new ej.Button($("#buttonType_reset"), {
size: "mini",
//this button is used to reset the form data to its initial value
type: "reset",
showRoundedCorner: true
});
});
});
```



```
}

```

Execute the above code to render the following output.



## Repeat Button

When you press button continuously, click event is raised at each specific time interval. This type of button is called **Repeat Button**. This functionality repeatedly raises the click event of button in both button click and from button in pressed state to the released state. **timeInterval** property is used to specify the time Interval for triggering click event, when the button is in pressed state. **repeatButton** property is used to set the button in repeat mode.

The following steps explains you the details about rendering the **Repeat Button**.

In the **HTML** page, add the following button elements to configure **Button** widget.

### HTML

```
<div class="control">
<div class="align">
<button id="button_repeat">Click
</button>
</div>
<div class="align">
<div><b>Event Trace</b></div>
<div class="eventTrace"></div>
</div>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ButtonComponent {
$(function () {
var basicButton = new ej.Button($("#button_repeat"), {
size: "mini",
showRoundedCorner: true,
//used to set the button in repeat mode
repeatButton: true,
//specifies the time interval for click method
//call, when the button is in pressed state
```

```

timeInterval: "200",
click: "btnClick"
});
});
}
//If the button is in pressed state or clicked, this method will be called
function btnClick(e) {
$(".eventTrace").html("click event has been triggered..<br>" +
$(".eventTrace").html());
}

```

Configure the **CSS** styles to apply on button

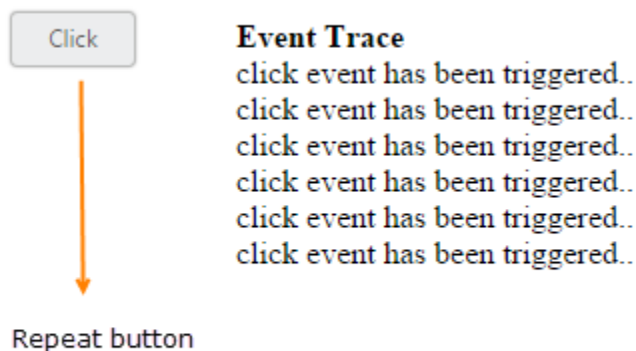
### CSS

```

<style>
.align {
display: table-cell;
padding-left: 50px;
}
</style>

```

Execute the above code to render the following output.



### Custom Buttons

Essential JavaScript Button control has an option of using normal ejButton as **Custom Button** to give visual weight to ejButton which helps user interface to notify and differentiate the priority action button with the other normal buttons.

Custom Buttons includes six predefined button styles and each button indicates unique sign of action to the user.

List of Custom Buttons

| Button Types | Class | Description |
|--------------|-------|-------------|
|--------------|-------|-------------|

|                    |           |   |
|--------------------|-----------|---|
| Primary Button     | e-primary | Provides extra visual weight and identifies the primary action in a set of buttons    |
| Link Button        | e-link    | Deemphasize a button by making it look like a link while maintaining button behavior. |
| Success Button     | e-success | Indicates a successful or positive action.  |
| Information Button | e-info    | Indicates a informative sign action to user   |
| Warning Button     | e-warning | Indicates the warning action.   |
| Danger Button      | e-danger  | Indicates the danger sign action to user.   |

To customize ejButton as anyone type of custom Buttons ,assign CSS class name of the custom button (For Ex:**e-success**) to cssClass property of ejButton.

The following steps explains you the details about customizing normal ejButton with above mentioned button types.

### HTML

```
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<div class="frame">
<div class="control">
<div class="btnsht">
<button id="PrimaryBtn">Primary</button>
</div>
<div class="btnsht">
<button id="DangerBtn">Danger</button>
</div>
<div class="btnsht">
<button id="InfoBtn">Information</button>
</div>
<div class="btnsht">
<button id="WarningBtn">Warning</button>
</div>
<div class="btnsht">
<button id="SuccessBtn">Success</button>
</div>
<div class="btnsht">
<button id="LinkBtn">Link</button>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```
module ButtonComponent {
  $(function () {
    var basicButton = new ej.Button($("#PrimaryBtn"), {
      size: "medium",
      showRoundedCorner: true,
      cssClass: 'e-primary'
    });
    var basicButton1 = new ej.Button($("#DangerBtn"), {
      showRoundedCorner: true,
      size: "medium",
      cssClass: 'e-danger'
    });
    var basicButton2 = new ej.Button($("#InfoBtn"), {
      showRoundedCorner: true,
      size: "medium",
      cssClass: 'e-info'
    });
    var basicButton3 = new ej.Button($("#WarningBtn"), {
      showRoundedCorner: true,
      size: "medium",
      cssClass: 'e-warning'
    });
    var basicButton4 = new ej.Button($("#LinkBtn"), {
      size: "medium",
      showRoundedCorner: true,
      cssClass: 'e-link'
    });
    var basicButton5 = new ej.Button($("#SuccessBtn"), {
      size: "medium",
      showRoundedCorner: true,
      cssClass: 'e-success'
    });
  });
}
```

### CSS

```
.frame {
margin: auto;
width: 400px;
}
.control .btnsht {
width: 125px;
display: inline-block;
}
.e-btn.e-select.e-btn-medium{
width:115px;
}
```

Execute the above code to render the following output.



### Image in Button

The **Essential Studio for JavaScript** has an option of using custom image in a button by assigning a CSS class with background-image to prefixIcon or suffixIcon property of ejButton. Please, refer the following below steps.

Create a custom CSS class with background-image property. Use the following syntax to apply class names.

**Syntax:** .e-icon .e-[icon name]

### CSS

```
.e-icon .e-profile{
background-image: url('profile.png');
}
```

### HTML

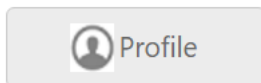
```
<button id="button">Profile</button>
```

Now, assign this custom CSS class name to prefixIcon or suffixIcon property of the ejButton.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ButtonComponent {
$(function () {
var basicButton = new ej.Button($("#button"), {
contentType: "textandimage",
prefixIcon: "e-icon e-profile",
size: "large",
showRoundedCorner: true
});
});
}
```

Execute the above code to render the following output.



### RTL Support

In some cases you can use right to left alignment. Here, RTL support is provided using **enableRTL** property. In RTL mode when you have more than one content (image/text, image/image) in button, then these content are aligned in right to left format. For example, when text is in left and image is in right position, after applying right to left alignment these position are interchanged.

The following steps explain the details about rendering the button with Right to left alignment support. In the **HTML** page, add the following button elements to configure button widget.

### HTML

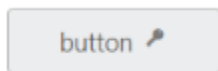
```
<button id="button_rtl">button</button>
```

### JAVASCRIPT

```
$(function () {
  $("#button_rtl").ejButton({
    size: "large", contentType: ej.ContentType.TextAndImage,
    showRoundedCorner: true,
    prefixIcon: "e-icon e-login",
    //used to enable or disable RTL support for button
    enableRTL: true
  });
});
```

In above mentioned code example prefixIcon property is used and the icon that is to be on left side, (before text) is rendered on right side as enableRTL property is used with prefixIcon. Consequently, the alignment is changed in right to left order.

Execute the above code to render the following output.



### Icons

The **Essential Studio for JavaScript** provide icons library that contains the number of in-built icons that can be applied for CSS class names to elements and refer “ej.widgets.all.core.min.css” file. Use the following syntax to apply class names.

**Syntax:** .e-icon .e-[icon description]

### HTML

```
.e-icon .e-search
```

### Adding icon in Button

For example, you can render the desired icon in the button by using the following table that contains the listed icons CSS class names in the “prefixIcon” property of button component. Also, use “contentType” property to display the icon in the button. In the following code example, specify the “**ContentType**” of the button as imageonly.

Refer to the following link to know what are the values passed in the “**ContentType**” property

<https://help.syncfusion.com/api/js/global#members:contenttype>

Also in the button sample, you can use the icon class names as follows,

### JAVASCRIPT

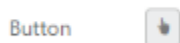
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```

module ButtonComponent {
  $(function () {
    var basicButton = new ej.Button($("#buttonId"), {
      contentType: "imageonly",
      prefixIcon: "e-icon e-handup"
    });
  });
}

```

Execute the above code to render the following output.






















Icon library used in button component

[List of Icons](#)








The complete list of icons is listed in the following table.



















List of icons




















|                        |   |
|------------------------|---|
| e-unpin                |    |
| e-pin                  |    |
| e-upload               |  |
| e-reload               |  |
| e-collapse             |  |
| e-cancel               |  |
| e-expand               |  |
| e-minimize             |  |
| e-login                |  |
| e-orientationlandscape |  |
| e-alignleft            |  |
| e-aligncenter          |  |

|                     |   |
|---------------------|---|
| e-alignright        | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-alignjustify      | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-alignnone         | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-filterset         | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-filternone        | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-arrowheadup-2x    | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-arrowheaddown-2x  | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-arrowheadleft-2x  | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-arrowheadright-2x | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-numbering         | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.  |
| e-bullets           | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-maximize          | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-delete            | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-scroll            | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-right-scroll      | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-search            | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-mediaback         | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-mediaforward      | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-medianext         | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |



|                           |  |
|---------------------------|--|
| e-mediaprev               | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file.   |
| e-mediaeject              | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file.   |
| e-mediaclose              | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file.   |
| e-mediapause              | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file.   |
| e-mediaplay               | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file.   |
| e-righttick               | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file.   |
| e-smile                   | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file.   |
| e-information             | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file.   |
| e-left-arrow              | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file.   |
| e-right-arrow             | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file.  |
| e-file-delete             | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file. |
| e-file-percentage-success | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file. |
| e-file-cancel             | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file. |
| e-file-percentage-failed  | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file. |
| e-file-retry              | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file. |
| e-resize-handle           | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file. |
| e-down-arrow              | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file. |
| e-time                    | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file. |
| e-up-arrow                | <br>The button image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file. |

|                  |   |
|------------------|---|
| e-date           | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-datetime       | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-collapse-arrow | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-expand-arrow   | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-restore        | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-plus           | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-minus          | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-handup         | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-clock          | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-cursor         | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.  |
| e-hyperlink      | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-hyperlinkbreak | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-settings       | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-shoppingcart   | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-palette        | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-warningmessage | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-cut            | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-copy           | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-paste          | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |

|                  |   |
|------------------|---|
| e-edit           | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-swapleft       | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-swaprightright | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-swaptop        | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-swaptopdown    | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-zoomin         | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-zoomout        | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-star           | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-home           | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.   |
| e-clipboard      | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.  |
| e-userlogin      | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-dataexport     | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-arrowheadright | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-arrowheaddown  | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-undo           | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-redo           | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-bold           | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-italic         | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |
| e-underline      | <br>The icon image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location. |

|                 |  |
|-----------------|--|
| e-strikethrough |  |
| e-font          |  |
| e-rarrowdown    |  |
| e-rarrowleft    |  |
| e-rarrowup      |  |
| e-rarrowright   |  |
| e-calender      |  |
| e-save          |  |

## Miscellaneous

### Text

You can display the user defined text for **Button**. Using **text** property, you can easily set text content for button. This text property overwrites the text that is provided on input button element.

The following steps explains you the details about rendering the button with specified text.

In the **HTML** page, add the following button elements to configure **Button** widget.

### HTML

```
<button id="button_text">button</button>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ButtonComponent {
$(function () {
var basicButton = new ej.Button($("#button_text"), {
size: "mini",
//used to set the text content for button
text: "Enter"
});
});
}
```

In the above code, the content of button “button” is replaced by the text value “Enter” that is given using text property.

Execute the above code to render the following output.



### Show Rounded Corner

Specifies the corner of button in round shape. By default button doesn't have rounded corner. To set rounded corner, you can enable **showRoundedCorner** property.

The following steps explains you the details about rendering the button with rounded corner.

In the **HTML** page, add the following button elements to configure **Button** widget.

#### HTML

```
<button id="button roundedCorner">button</button>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ButtonComponent {
$(function () {
var basicButton = new ej.Button($("#button_roundedCorner"), {
size: "mini",
//Enable or disable the rounded corner for button
showRoundedCorner: true
});
});
}
```

Execute the above code to render the following output.



## Chart

### Overview

The Typescript Chart control is a visually stunning Charting component. It includes common Chart types ranging from line Charts to specialized financial Charts.

Some of the key features are,

- Supports highly interactive 28 chart types; 8 charts can be viewed in 3D view.
- Data Binding with local and remote data source.

- Supports multiple axes, and able to plot data with different data types such as numbers, date time and strings.
- Supports interactive features like zooming, panning, crosshair, trackball, tooltip and data point selection.
- Supports 10 types of technical indicators and trend lines.
- Easily customize each and every element of Chart.

## Getting Started

For common getting started of typescript , you can refer [here](#).

The default type definition file ej.web.all.d.ts needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the jquery.d.ts file in your TypeScript application, which can be downloaded from [here](#).

## Adding Script Reference

Create an **HTML** page and add the scripts references in the order mentioned in the following code example.

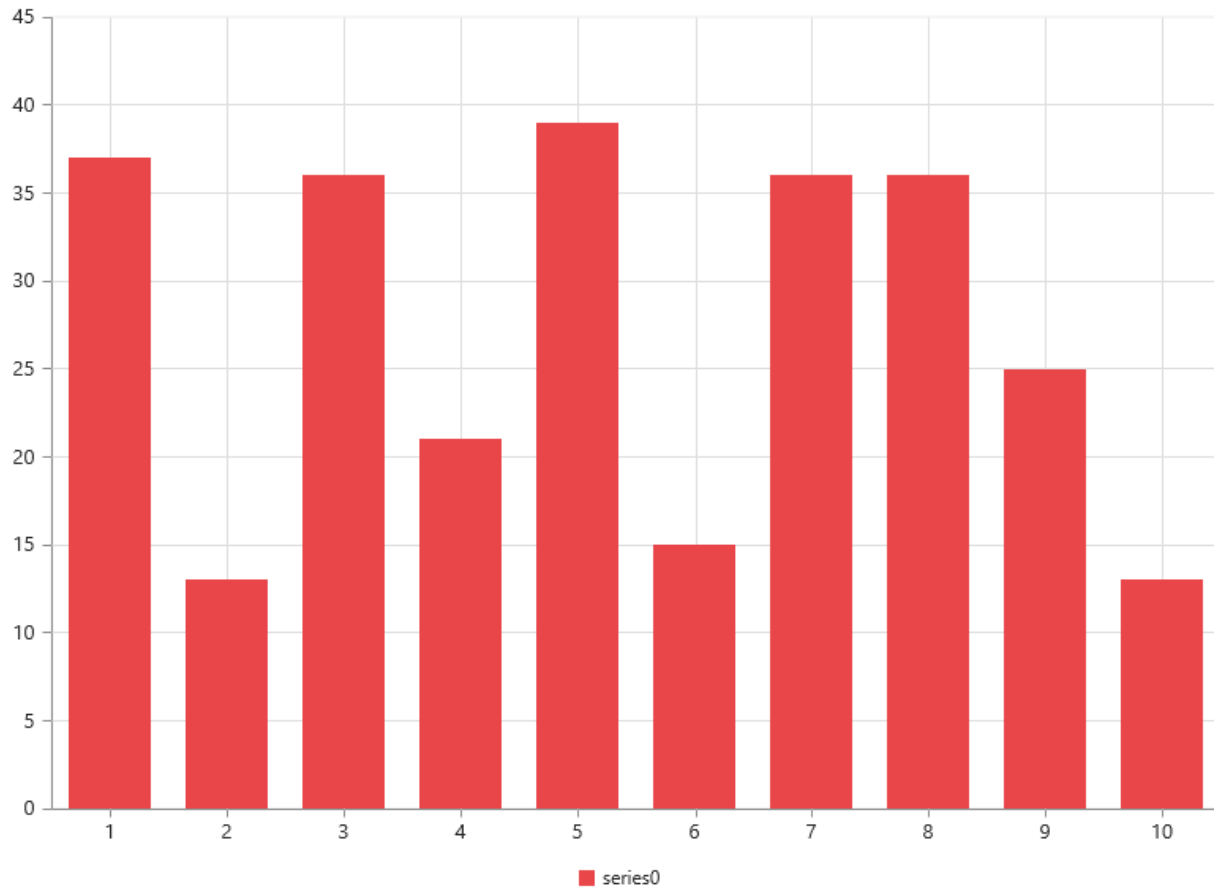
### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

In the above code, `ej.web.all.min.js` script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

## Create your chart

In this tutorial, you will learn how to create a simple chart. The following screen shot displays the output after completing this tutorial.



1.Create a  
tag.

#### HTML

```
<html> <body> <div id="Chart"></div> </body> </html>
```

2.Initialize the Chart in ts file by using the `ej.Chart` method.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
  $(function () {
    var chartsample = new ej.datavisualization.Chart($("#Chart"));
  });
}
```

Now, the Chart is rendered with some auto-generated random values and with default Column chart type.

Initialize the chart by using the `ejChart` method. The chart is rendered to the size of its container, by default. You can also customize the chart dimension either by setting the width and height of the container element as in the above code example or by using the **Size** option of the Chart.

### Populate chart with data

Now, this section explains how to plot JSON data to the Chart. First, let us prepare a sample JSON data with each object containing following fields – month and sales.

#### JAVASCRIPT

```
var chartData = [
  { month: 'Jan', sales: 35 },
  { month: 'Feb', sales: 28 },
  { month: 'Mar', sales: 34 },
  { month: 'Apr', sales: 32 },
  { month: 'May', sales: 40 },
  { month: 'Jun', sales: 32 },
  { month: 'Jul', sales: 35 },
  { month: 'Aug', sales: 55 },
  { month: 'Sep', sales: 38 },
  { month: 'Oct', sales: 30 },
  { month: 'Nov', sales: 25 },
  { month: 'Dec', sales: 32 }];
```

Add a Series to the Chart using **Series** option and set the chart type as **Line** using **type** option.

#### JAVASCRIPT

```
$(function () {
  var chartsample = new ej.datavisualization.Chart($("#Chart"), {
    series:
    [
      {
        type: "line"
      }
    ]
  });
});
```

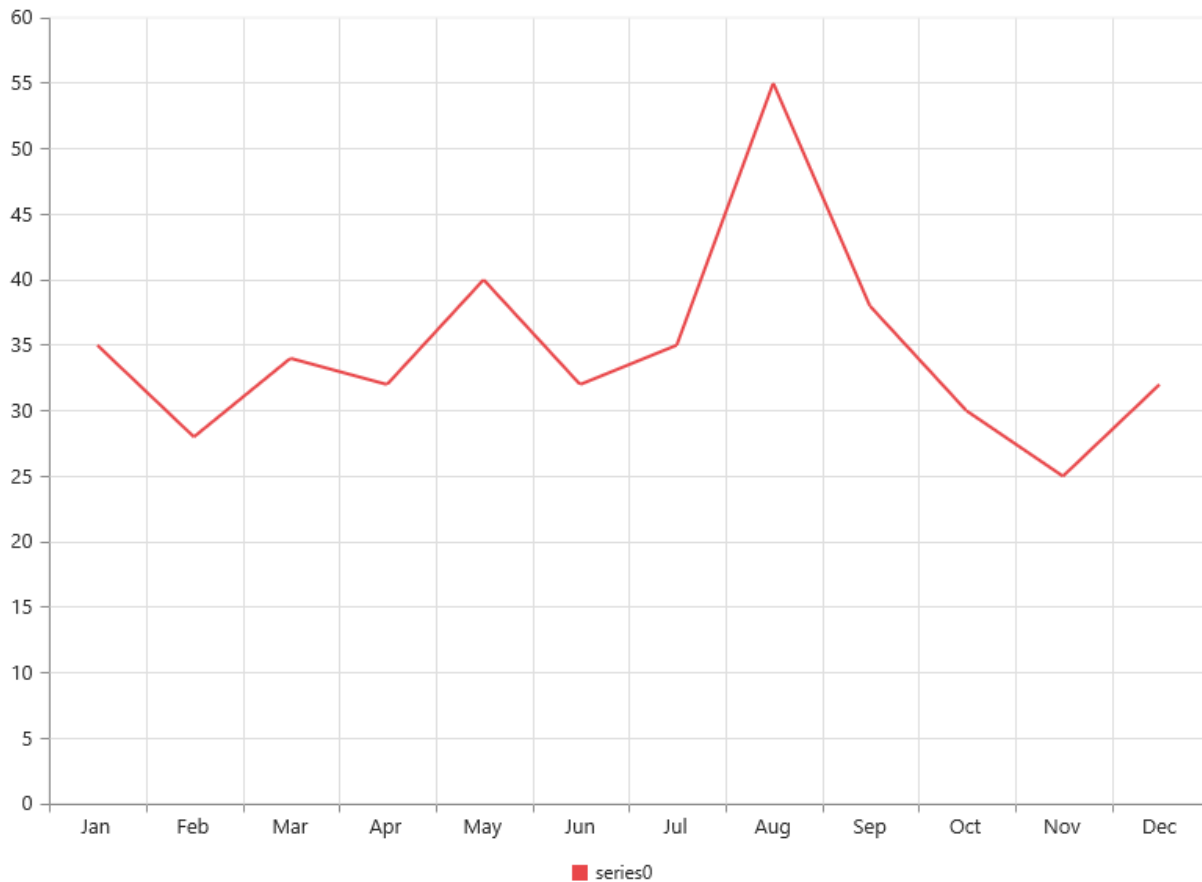
You can also add multiple series tags based on your requirement.

Next, map the Month and Sales values in the data source to the Line series by setting XName and YName with the field names respectively, and then set the actual data using DataSource option.

#### JAVASCRIPT

```
$(function () {
  var chartsample = new ej.datavisualization.Chart($("#Chart"), {
    series:
    [
      {
        dataSource: chartData,
        xName: "month",
        yName: "sales",
      },
    ],
    size: {height: "400", width: "600"}
  });
});
```



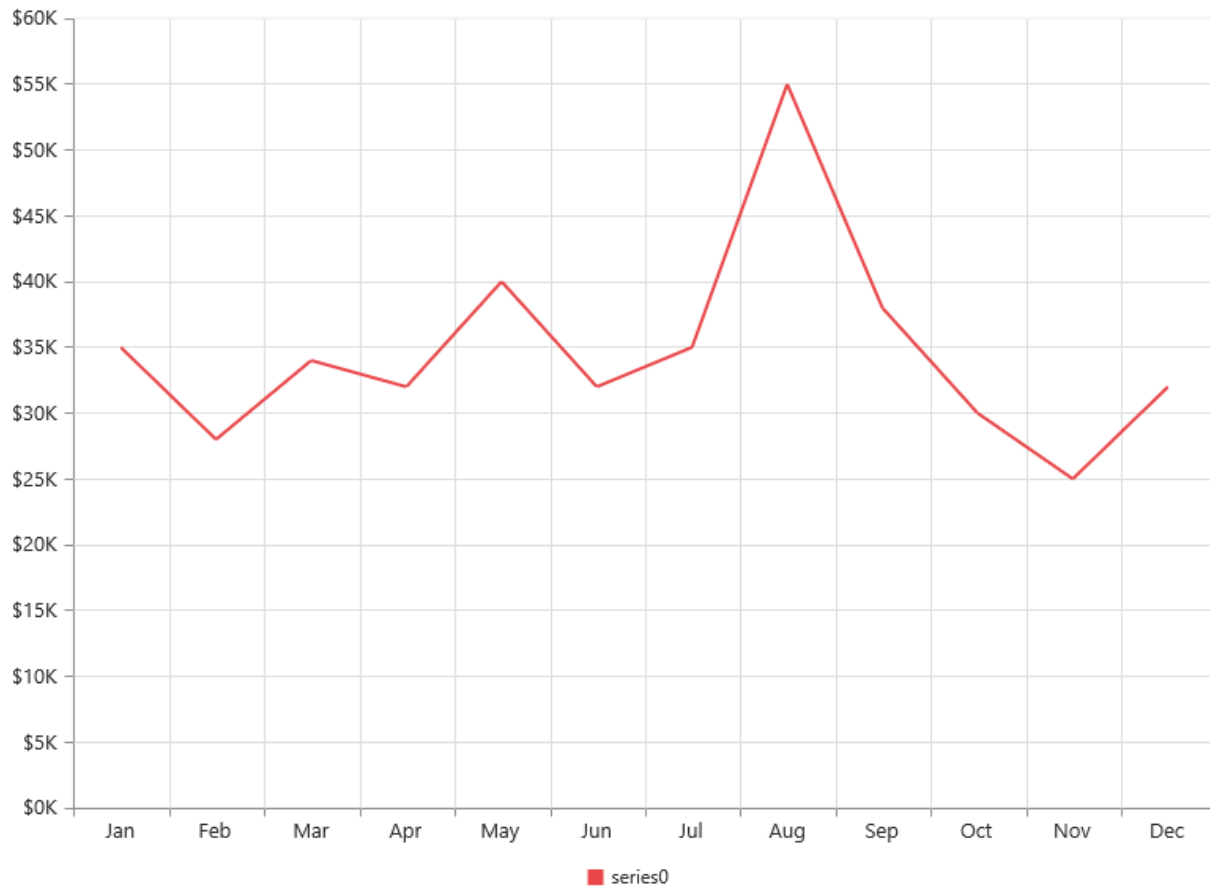


Since the data is related to Sales, format the vertical axis labels by adding '\$' as a prefix and 'K' as a suffix to each label. This can be achieved by setting the "\${value}K" to the **labelFormat** option of the axis. Here, {value} acts as a placeholder for each axis label, "\$" and "K" are the actual prefix and suffix added to each axis label.

The following code example illustrates this,

#### JAVASCRIPT

```
$(function () {  
    var chartsample = new ej.datavisualization.Chart($("#Chart"), {  
        primaryYAxis: {  
            labelFormat: "{value}k"  
        }  
    });  
});
```



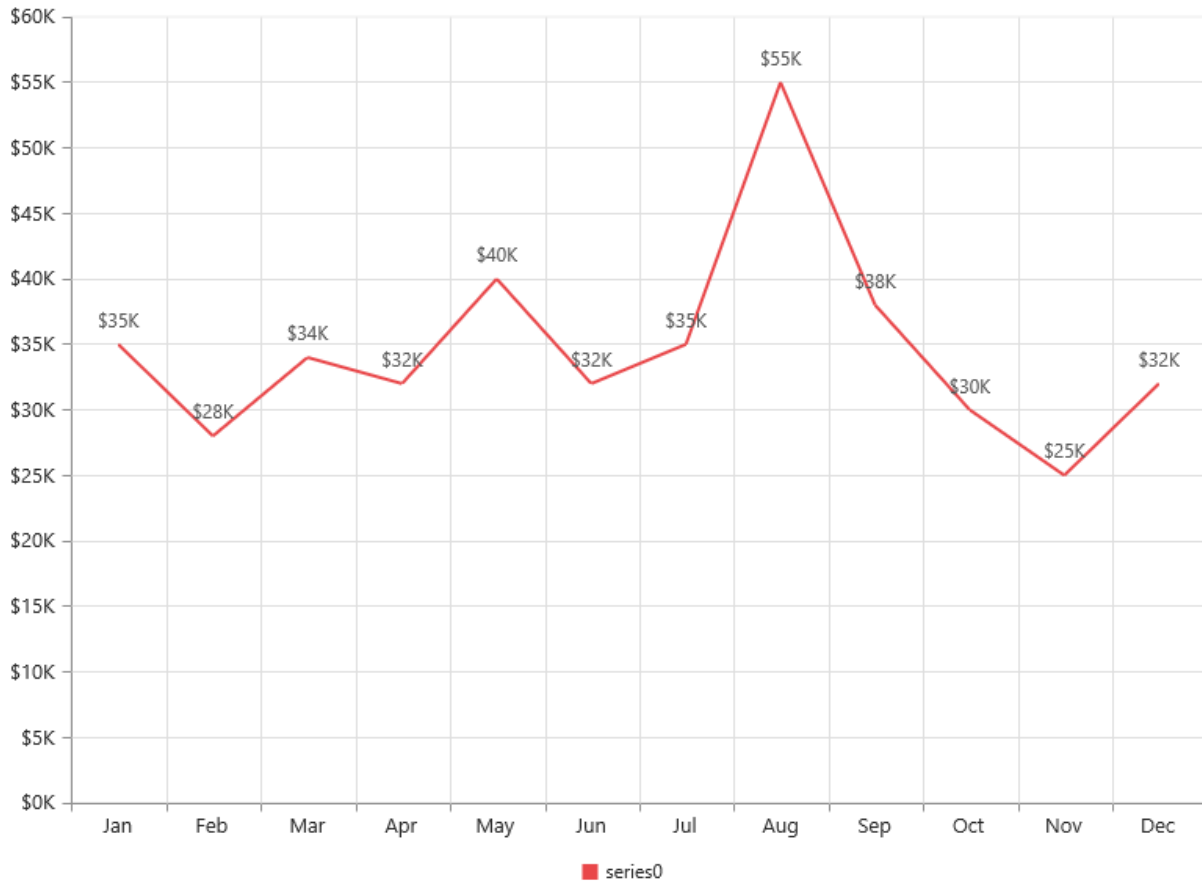
### Add Data Labels

You can add data labels to improve the readability of the chart. This can be achieved by enabling the Visible option in the **dataLabel** option. Now, the data labels are rendered at the top of all the data points.

The following code example illustrates this,

### JAVASCRIPT

```
$(function () {  
  var chartsample = new ej.datavisualization.Chart($("#Chart"), {  
    series:  
    [  
      {  
        marker: {  
          dataLabel: {  
            //Enable data label in the chart  
            visible: true  
          }  
        }  
      }  
    ]  
  });  
});
```



There are situations where the default label content is not sufficient to the user. In this case, you can use the **template** option to format the label content with some additional information.

### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="dataLabelTemplate" style="display:none; padding:3px;background-
color:#B9C5C9; opacity:0.8;">
<div id="point">#point.x#:$#point.y#K</div>
</div>
</body>
</html>
```

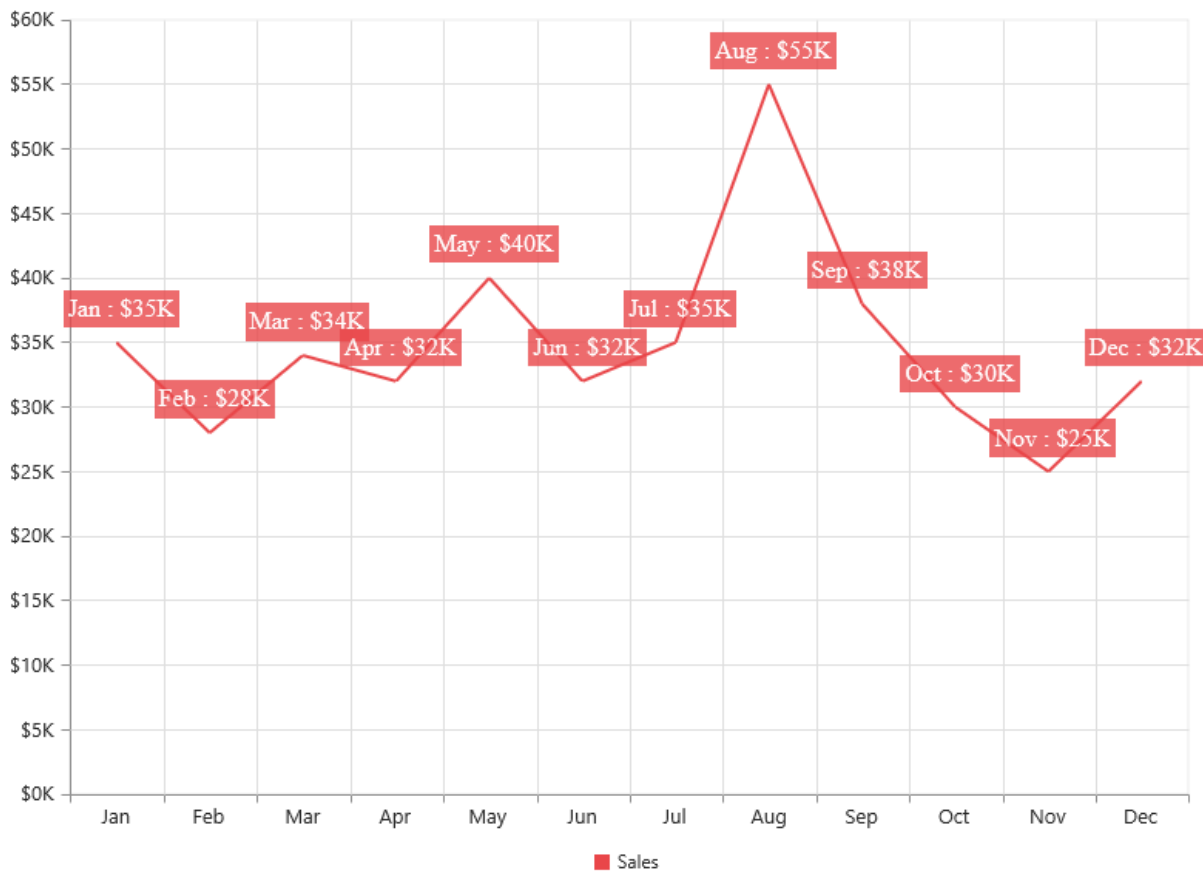
The above HTML template is used as a template for each data label. Here, “point.x” and “point.y” are the placeholder text used to display the corresponding data point’s x & y value.

The following code example shows how to set the id of the above template to dataLabel template option,

### JAVASCRIPT

```
$(function () {
var chartsample = new ej.datavisualization.Chart($("#Chart"), {
series:
```

```
[
{
marker: {
dataLabel: {
visible: true,
//Set the id of HTML template to the chart series
template: "dataLabelTemplate"
} }
}
]
});
});
```



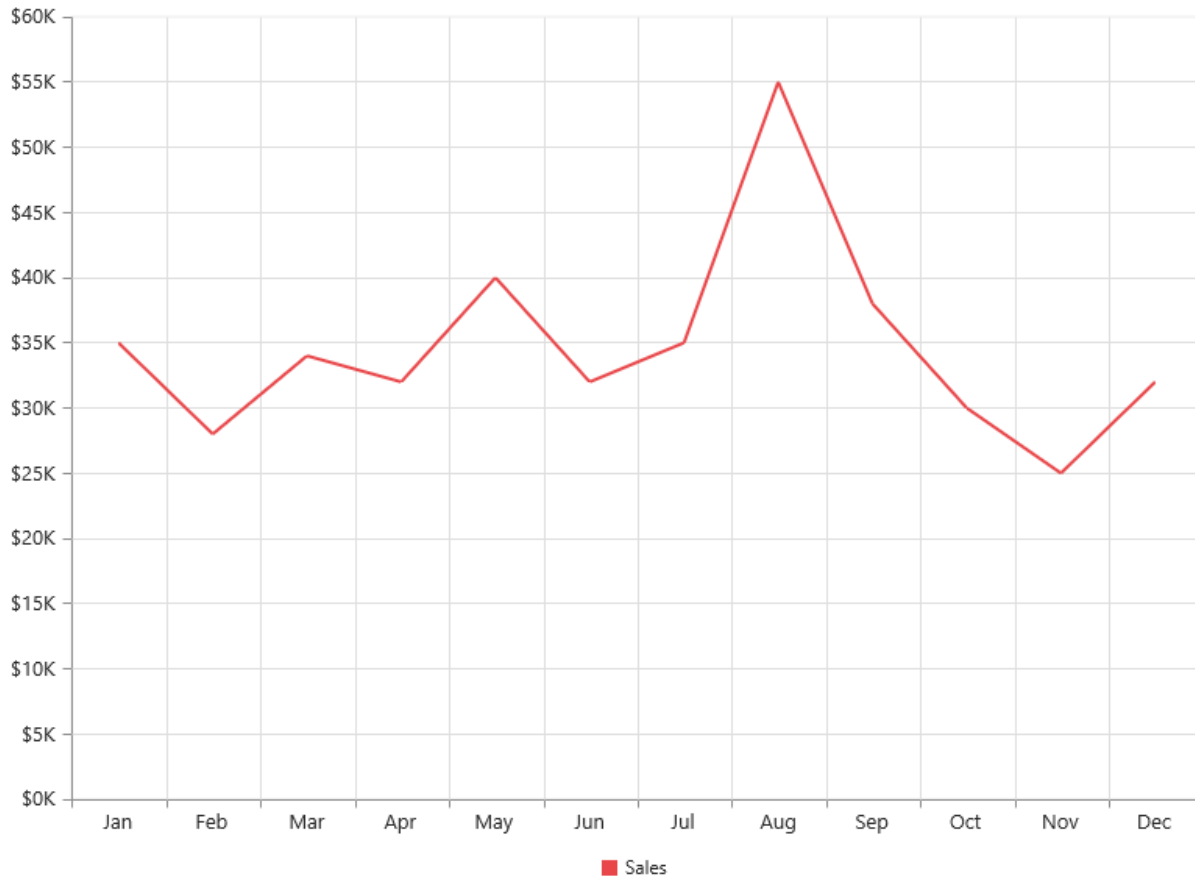
### Enable Legend

You can enable or disable the legend by using the Visible option in the **legend** option. It is enabled in the chart, by default.

### JAVASCRIPT

```
$(function () {
var chartsample = new ej.datavisualization.Chart($("#Chart"), {
//Initializing Series
series: [{
//Add series name to display on the legend item
name: "Sales"
}],
});
```

```
legend: {  
  //Enable chart legend  
  visible: true  
};  
});
```



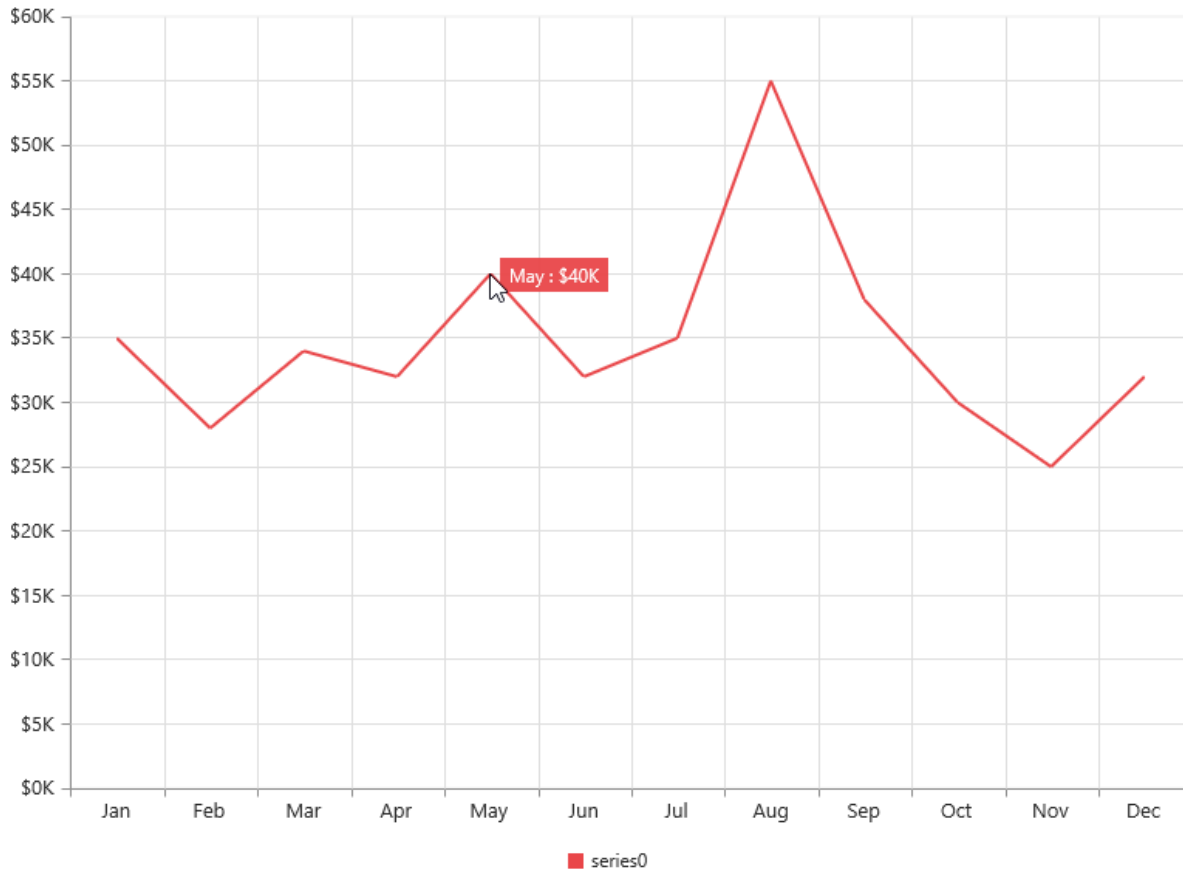
### Enable Tooltip

The Tooltip is useful when you cannot display information by using the **dataLabel** due to the space constraints. You can enable tooltip by using the Visible option of the **tooltip** option in the specific series.

The following code example illustrates this,

### JAVASCRIPT

```
$(function () {  
  var chartsample = new ej.datavisualization.Chart($("#Chart"), {  
    //Initializing Series  
    series: [{  
      //Enable tooltip in chart area  
      tooltip: {visible: true}  
    }],  
  });  
});
```

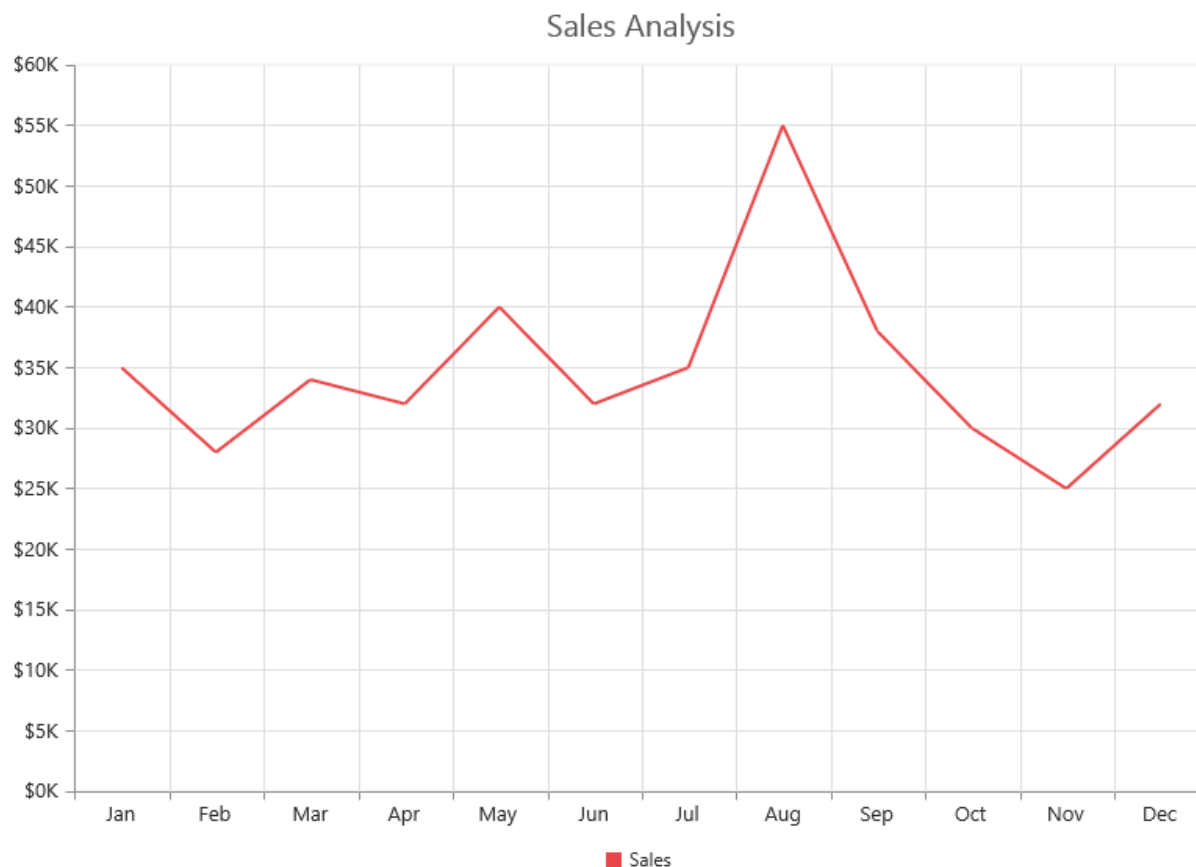


### Add Chart Title

You need to add a title to the chart to provide quick information to the user about the data being plotted in the chart. You can add it by using the text option of the **title** option.

### JAVASCRIPT

```
$(function () {  
  var chartsample = new ej.datavisualization.Chart($("#Chart"), {  
    title: {  
      //Add chart title  
      text: 'Sales Analysis'  
    },  
  });  
});
```



## Working with Data

### Local Data

There are two ways to provide local data to chart.

1. You can bind the data to the chart by using the [dataSource](#) property of the series and then you need to map the X and Y value with the [xName](#) and [yName](#) properties respectively.

**Note:** For the **OHLC** type series, you have to map four [dataSource](#) fields ([high](#), [low](#), [open](#) and [close](#)) to bind the data source and for the **bubble** series you have to map the [size](#) field along with the [xName](#) and [yName](#).

### JAVASCRIPT

```

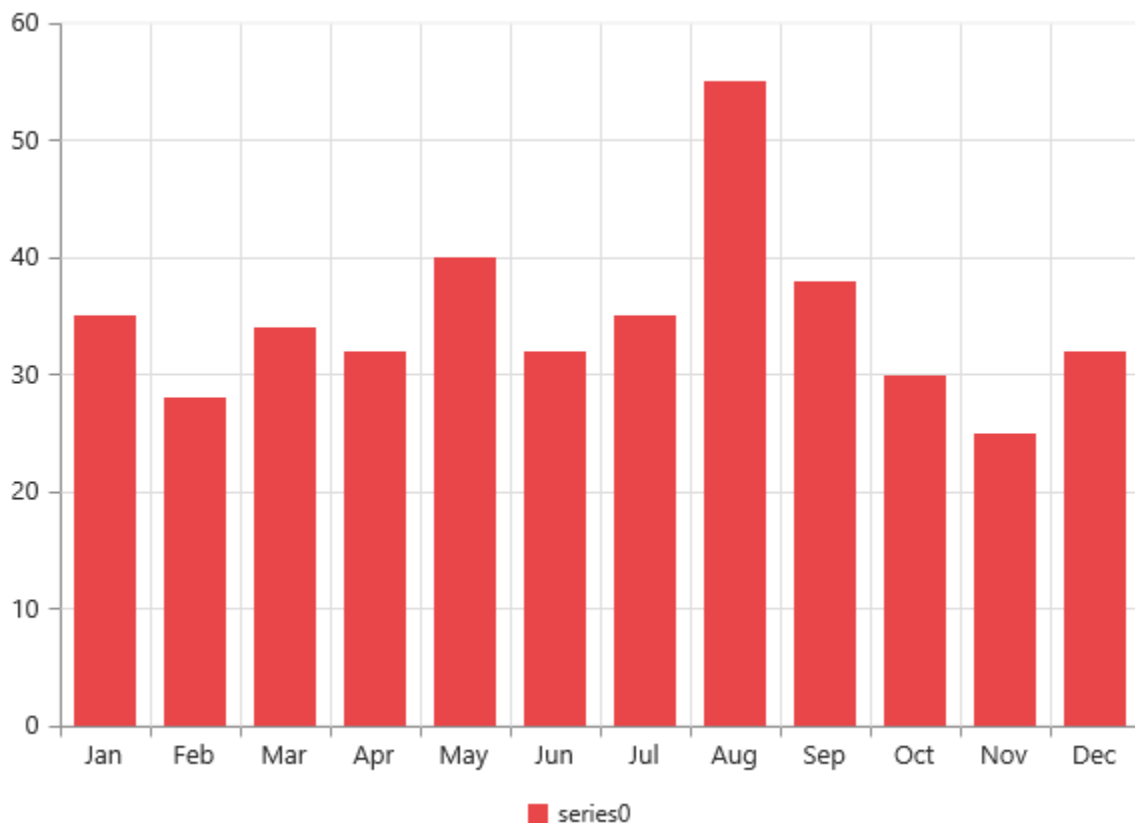
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
var chartData = [
  { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 }, { month: 'Mar',
sales: 34 },
  { month: 'Apr', sales: 32 }, { month: 'May', sales: 40 }, { month: 'Jun',
sales: 32 },
  { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 }, { month: 'Sep',
sales: 38 },
  { month: 'Oct', sales: 30 }, { month: 'Nov', sales: 25 }, { month: 'Dec',
sales: 32 }];
$(function () {

```

```

var chartsample = new ej.datavisualization.Chart($("#Chart"), {
  series: [{
    // ...
    //Add datasource and set xName and yName
    dataSource: chartData,
    xName: "month",
    yName: "sales"
  }]
  // ...
});

```



2.You can also plot data to chart using [points](#) option in the series. Using this property you can customize each and every point in the data.

### JAVASCRIPT

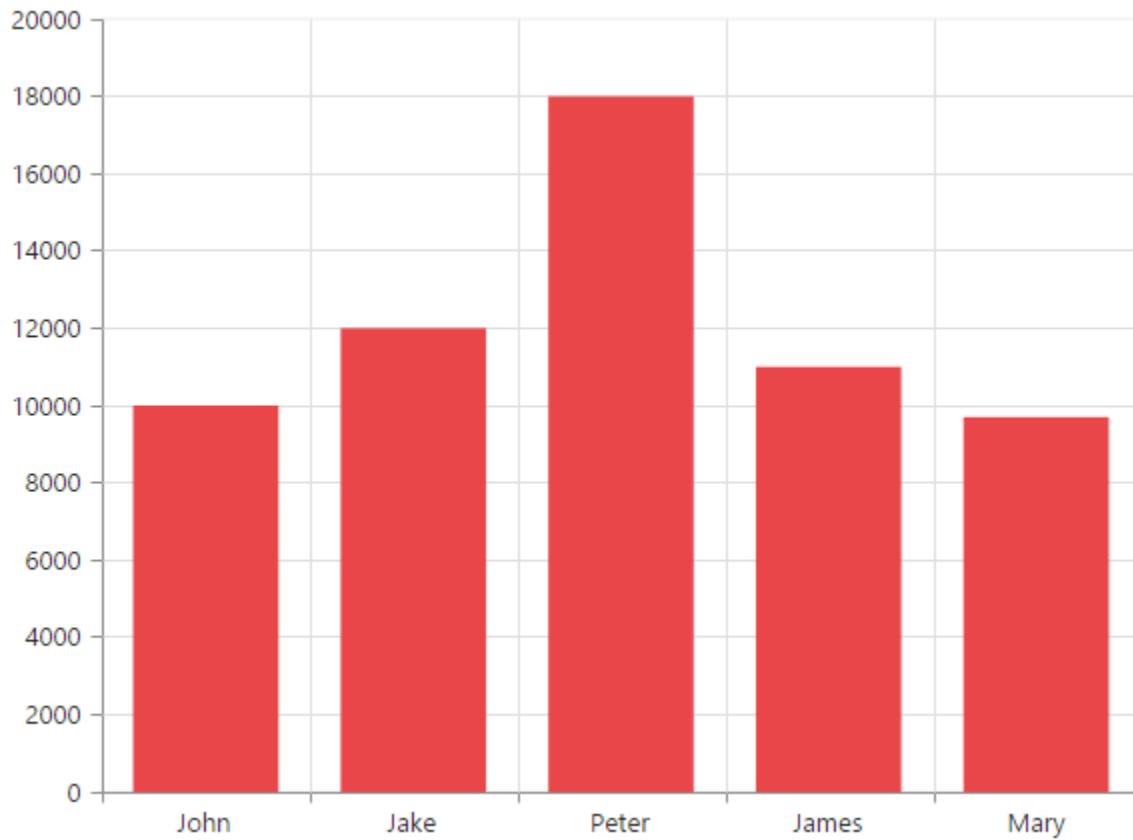
```

$(function () {
  var chartsample = new ej.datavisualization.Chart($("#Chart"), {
    // ...
    //Initializing Series
    series: [{
      //Adding data points using x and y field of points
      points: [{ x: "John", y: 10000 }, { x: "Jake", y: 12000 }, { x: "Peter", y: 18000 },
        { x: "James", y: 11000 }, { x: "Mary", y: 9700 }],
    }],
  });

```



```
// ...  
}],  
// ...  
});  
});
```



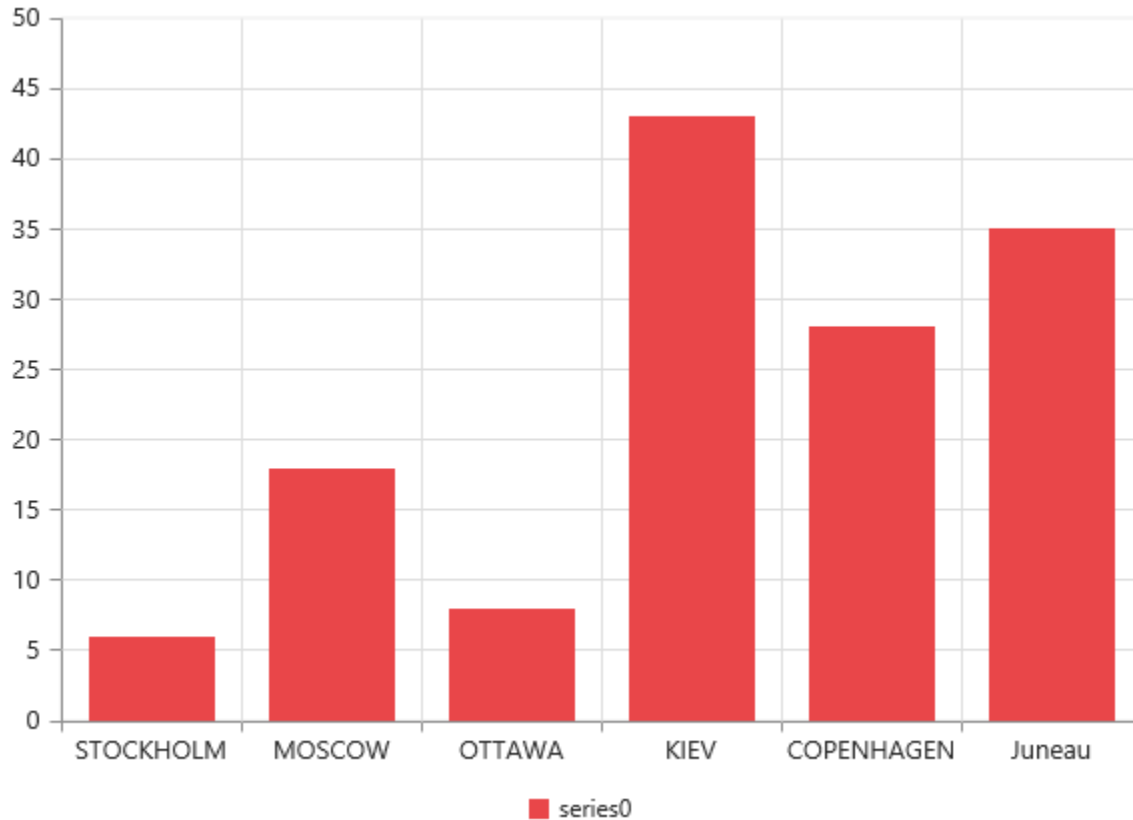
### Remote Data

You can bind the remote data to the chart by using the DataManager and you can use the [query](#) property of the series to filter the data from the dataSource.

### JAVASCRIPT

```
//Remote URL  
var dataManger = new ej.DataManager({  
  url: "http://mvc.syncfusion.com/Services/Northwnd.svc/"  
});  
// Query creation  
var query = ej.Query().from("Orders").take(6);  
$(function () {  
  var chartsample = new ej.datavisualization.Chart($("#Chart"), {  
    series: [{  
      type: 'column',  
      dataSource: dataManger,  
      xName: "ShipCity",  
      yName: "Freight",  
      query: query,  
    }],  
  });  
});
```

```
}},  
});  
});
```



### Chart Dimensions

You can set the size of the chart directly on the chart or to the container of the chart. When you do not specify the size, it takes 450px as the height and window size as its width, by default.

#### Set size for the container

You can customize the chart dimension by setting the width and height for the container element.

#### HTML

```
<div id="container" style="width:820px; height:500px;"></div>
```

#### JAVASCRIPT

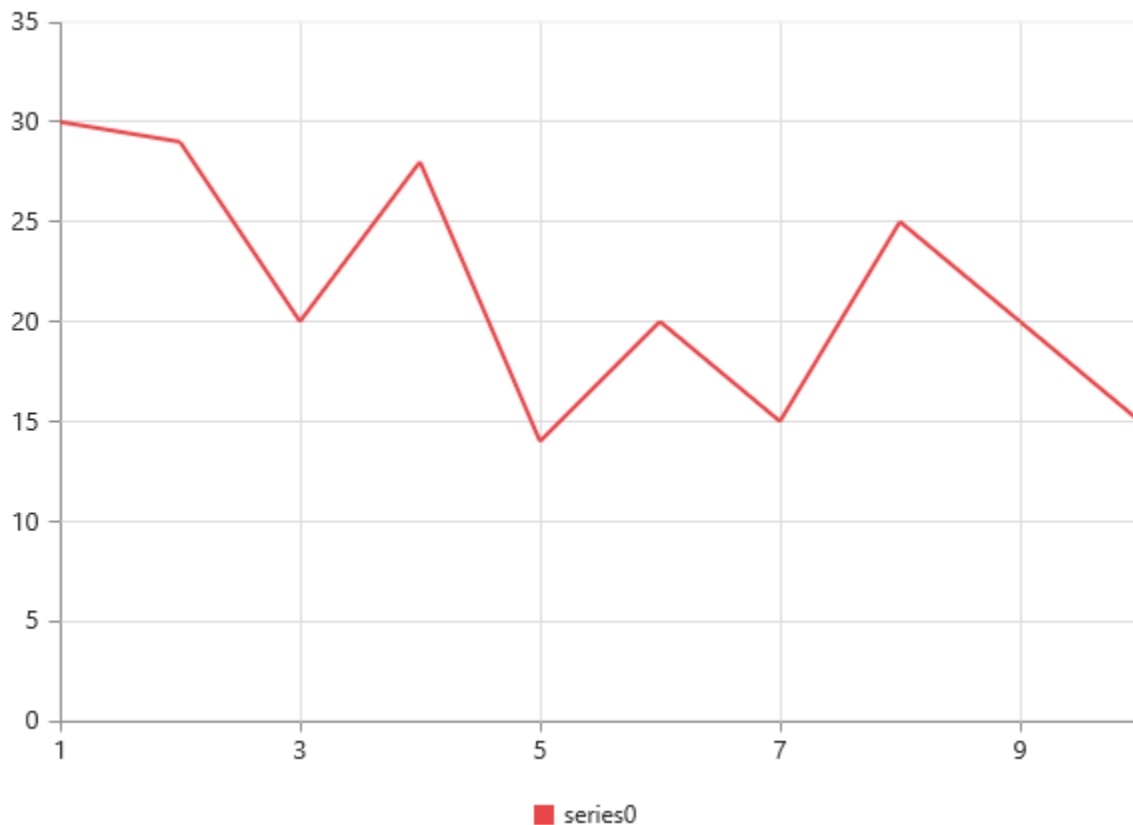
```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module ChartComponent {  
    $(function () {  
        var chartsample = new ej.datavisualization.Chart($("#container"), {  
        });  
    });  
}
```

### Set size in pixels

You can also set the chart dimension by using the [size](#) property of the chart.

#### JAVASCRIPT

```
$(function () {  
  var chartsample = new ej.datavisualization.Chart($("#container"), {  
    // ...  
    //Set size to chart container  
    size: { width: '600', height: '450' },  
  });  
});
```



### Setting size relative to the container size

You can specify the chart size in percentage by using the [size](#) property. The chart gets its dimension with respect to its container.

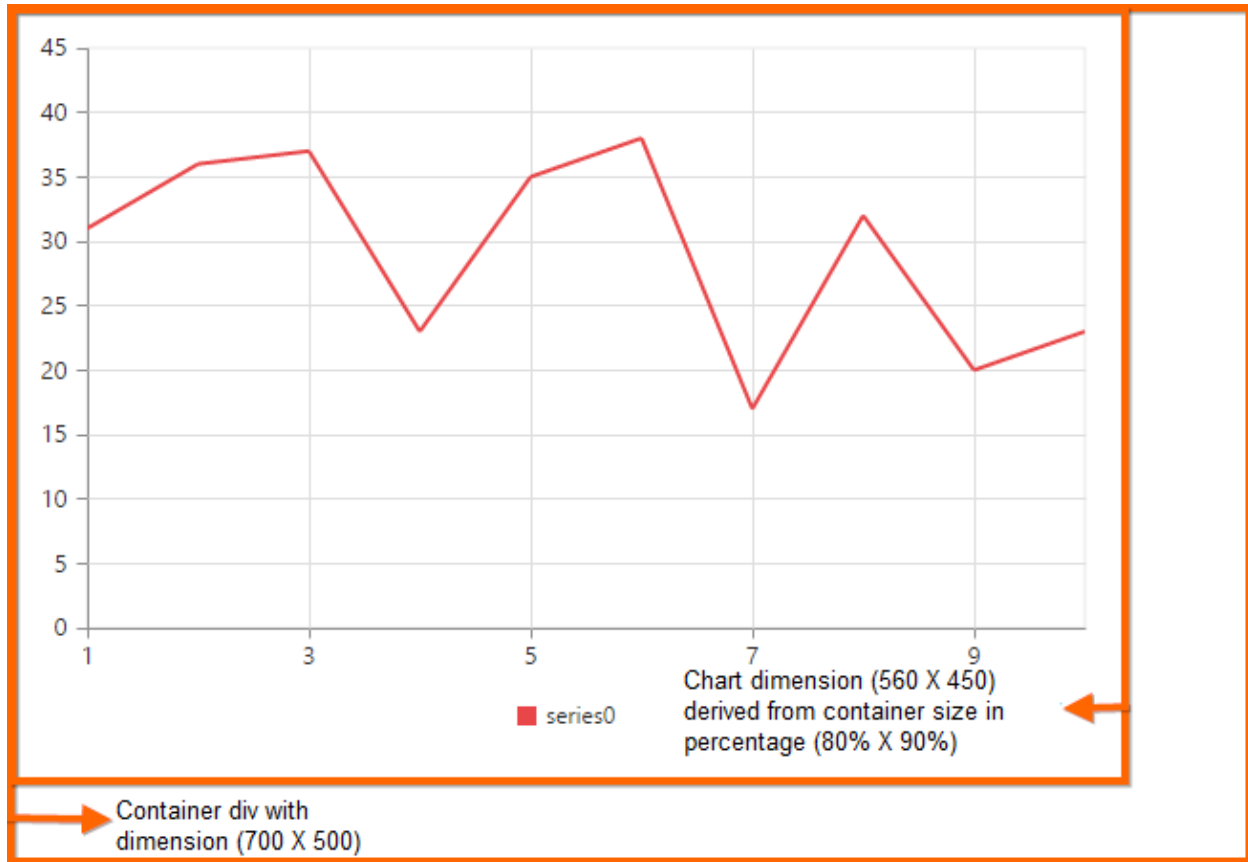
#### HTML

```
<div id="container" style="width:700px; height:500px"></div>
```

#### JAVASCRIPT

```
$(function () {  
  var chartsample = new ej.datavisualization.Chart($("#container"), {  
    // ...  
  });  
});
```

```
//Set size in percentage to chart container
size: { width: '80%', height: '90%' },
});
});
```



### Responsive chart

To resize the Chart when the browser or the chart container is resized, set the [isResponsive](#) property to **true**, where the chart adapts to the changes in size of the container.

### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#container"), {
// ...
//Enable isResponsive to change the chart size dynamically.
isResponsive: true
// ...
});
```

### Axis

**Charts** typically have two axes that are used to measure and categorize data: a vertical (y) axis, and a horizontal (x) axis.

Vertical axis always uses numerical or logarithmic scale. Horizontal(x) axis supports the following types of scale:

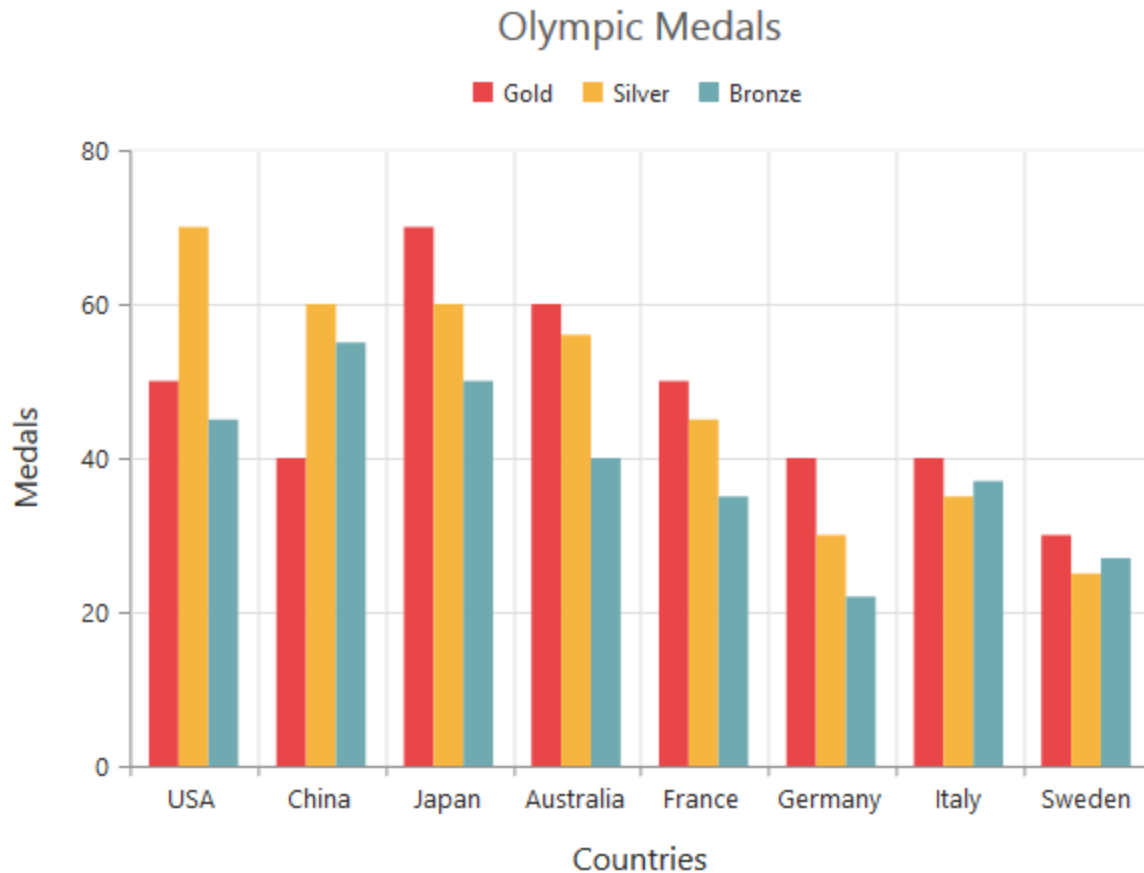
- Category
- Numeric
- DateTime
- DateTime Category
- Logarithmic

### Category Axis

Category axis displays the text labels instead of numbers. To use the categorical axis, you can set the [valueType](#) property of the axis to the **category**. Default value of [valueType](#) is **double**.

### JS

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ChartComponent {
  $(function () {
    var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
      primaryXAxis: {
        //Use categorical scale in primary X axis
        valueType: 'category',
        // ...
      },
      // ...
    });
  });
}
```

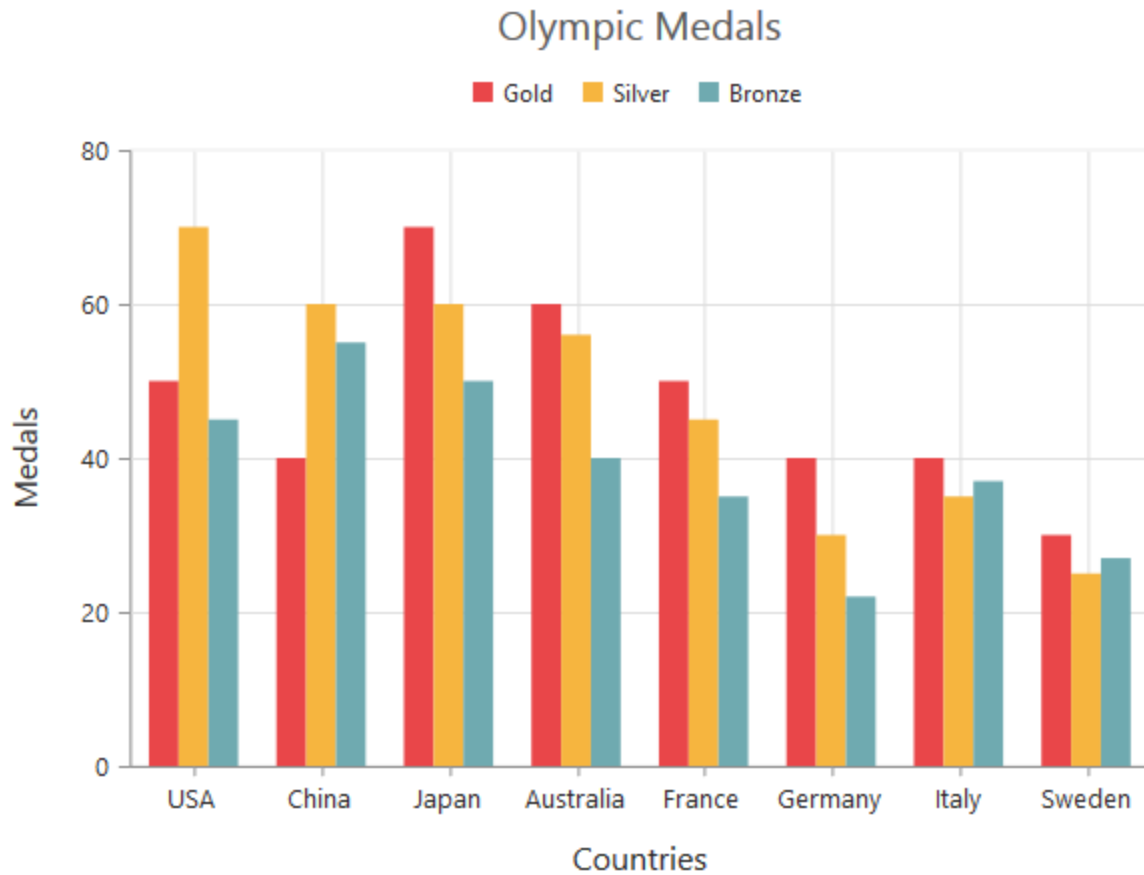


#### Place labels on ticks

Labels in the category axis can be placed on the ticks by setting the [labelPlacement](#) property of axis to the **onTicks**. The default value of the [labelPlacement](#) property is **betweenTicks** i.e. labels are placed between the ticks, by default.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Placing X-axis labels on the ticks
    labelPlacement: 'onTicks',
    // ...
  },
  // ...
});
```

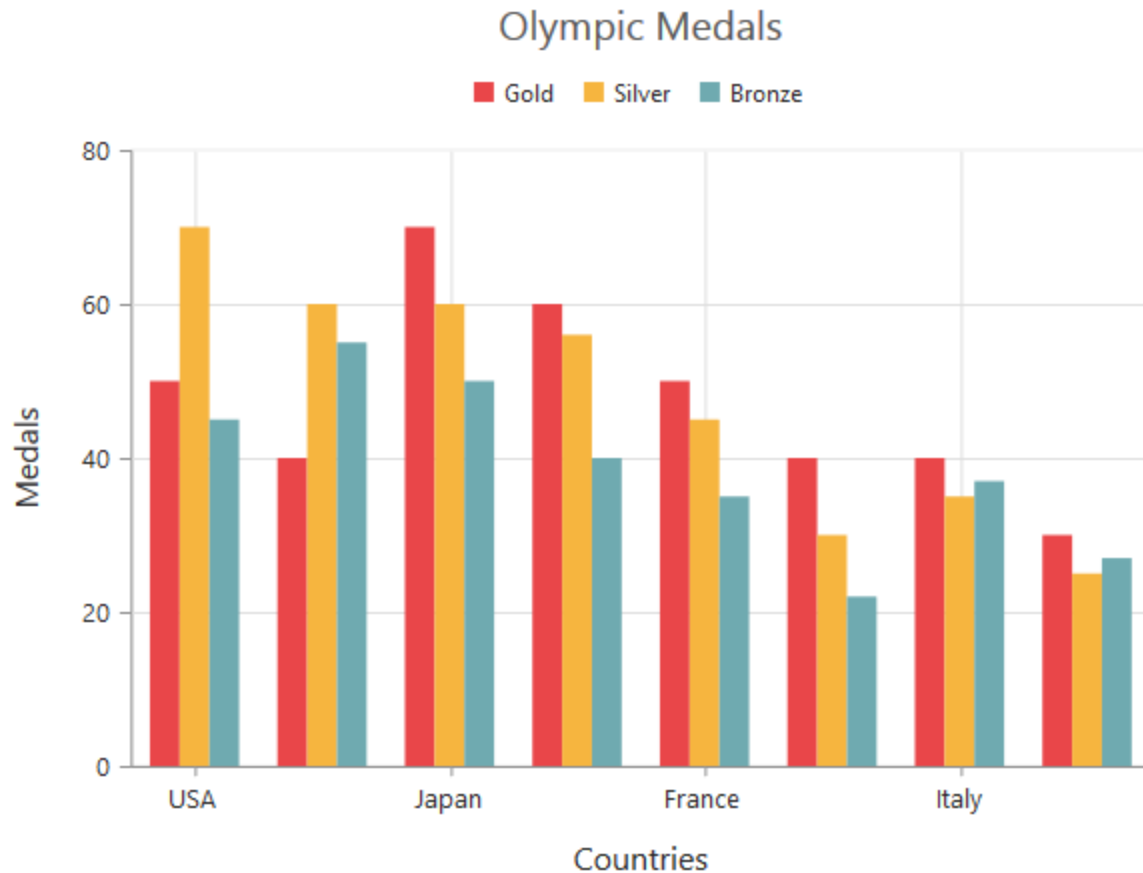


#### Display labels after a fixed interval

To display the labels after a fixed interval *n*, you can set the [interval](#) property of the axis range as *n*. The default value of the interval is 1 i.e. all the labels are displayed.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Displaying labels after 2 intervals
    range: { interval: 2 },
    // ...
  },
  // ...
});
```



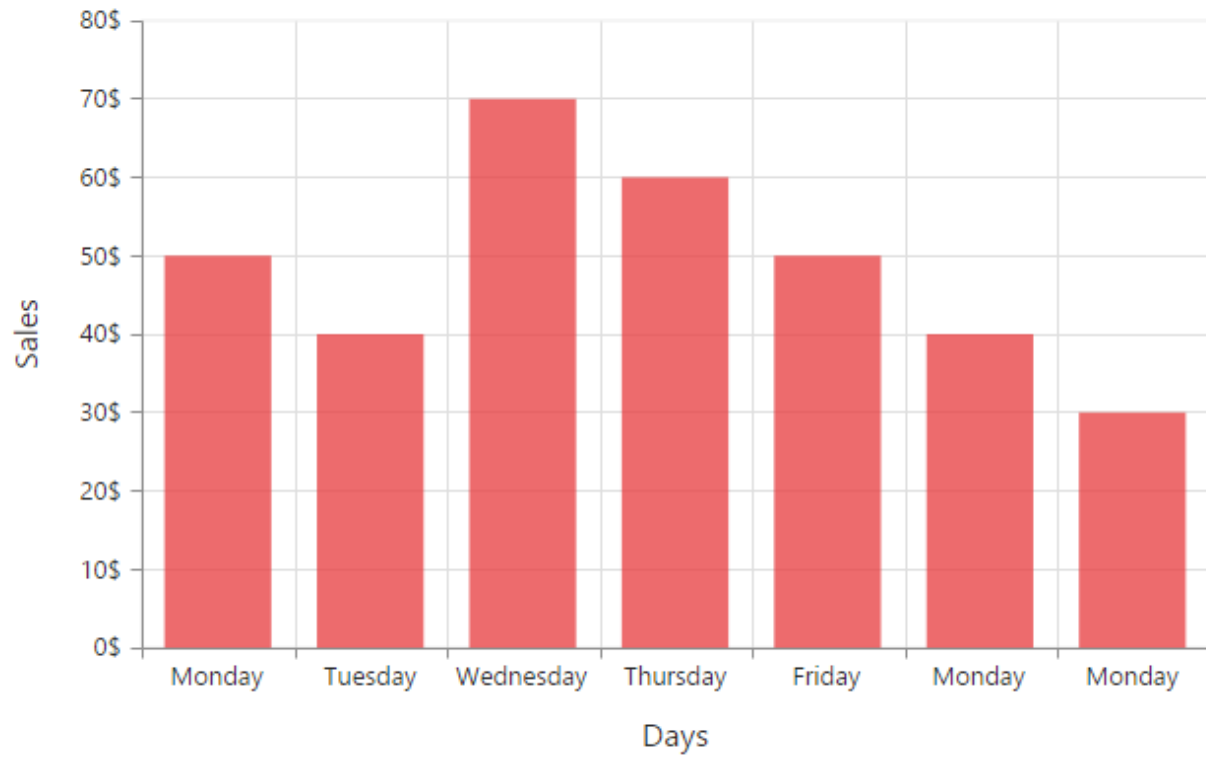
#### *Indexed Category Axis*

Category axis can also plot points based on index value of data points. Index based plotting can be enabled by setting [isIndexed](#) property to true in the axis.

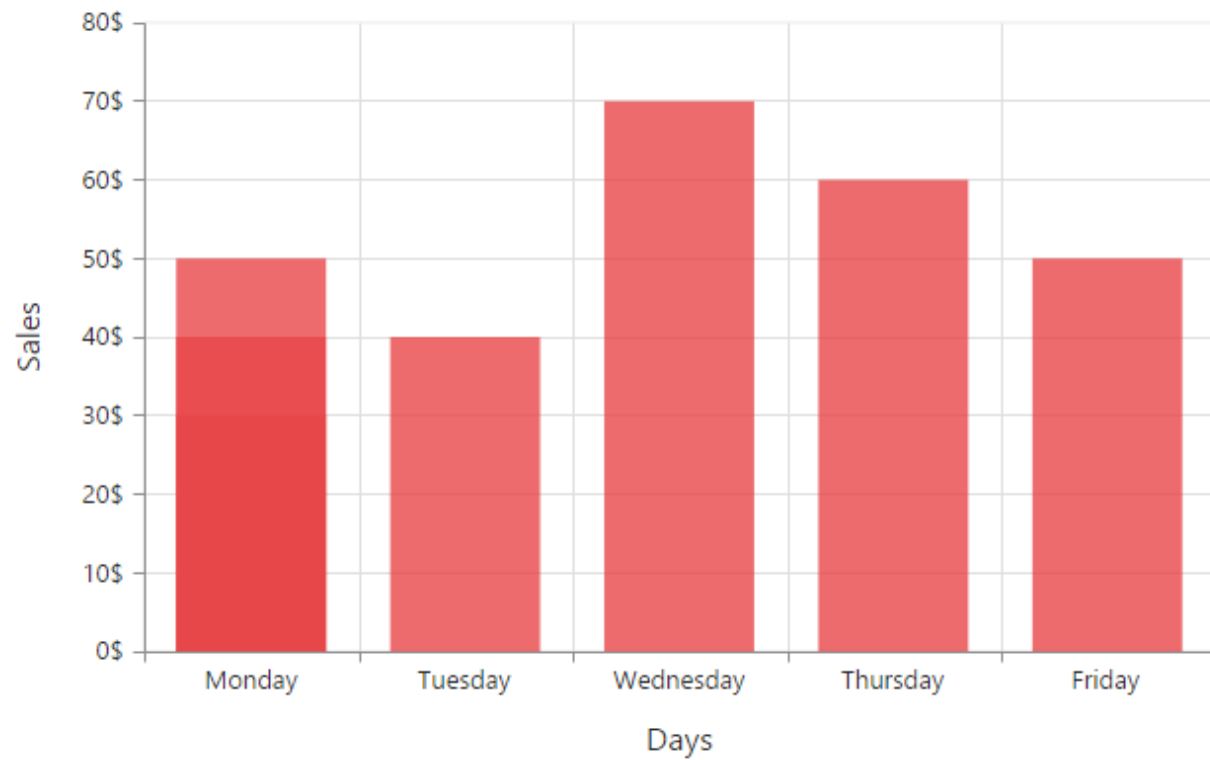
#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  primaryXAxis: {
    isIndexed: true
  },
  series:[{
    points:[{ x: "Monday", y: 50 }, { x: "Tuesday", y: 40 }, { x: "Wednesday",
    y: 70 },
    { x: "Thursday", y: 60 }, { x: "Friday", y: 50 },
    { x: "Monday", y: 40 }, { x: "Monday", y: 30 }
  ]
  }],
  // ...
});
```





While Category axis is Indexed value false

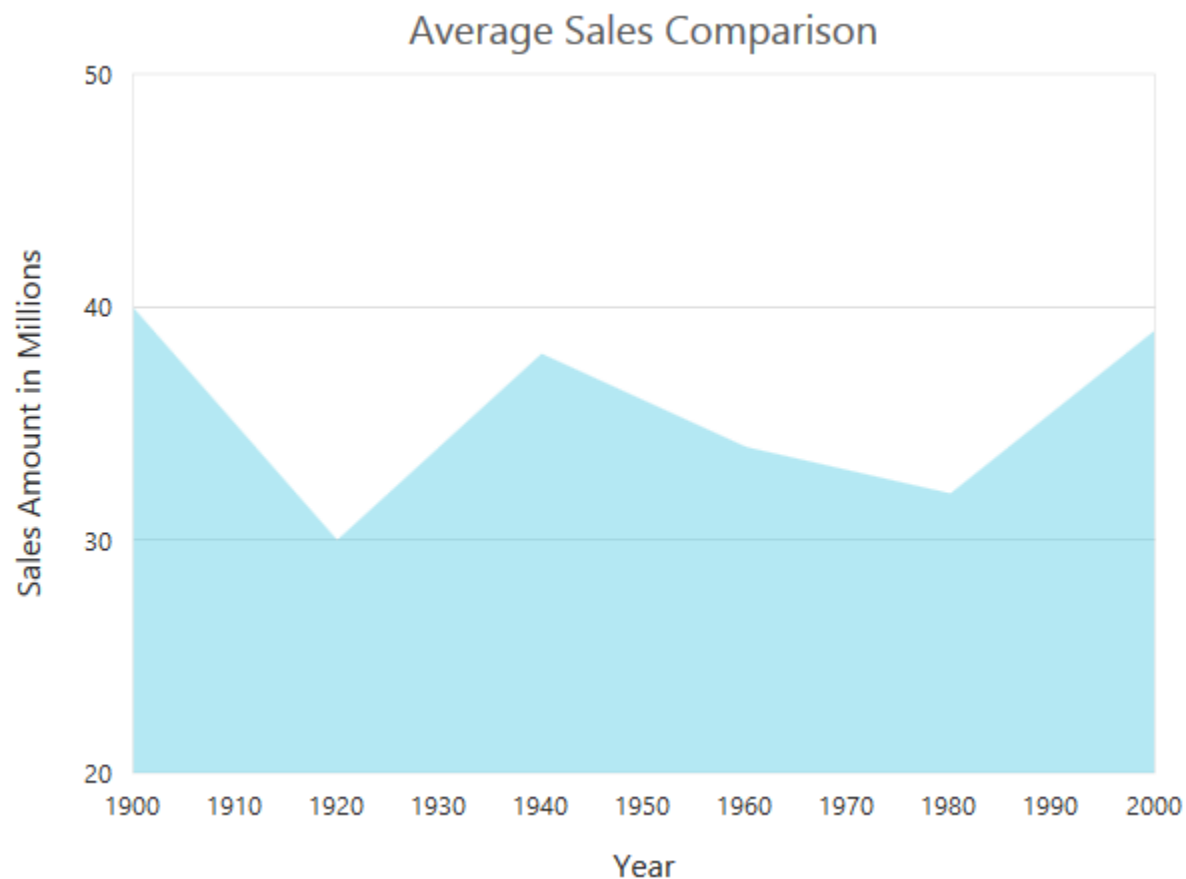


### Numeric Axis

Numeric axis uses numerical scale and displays numbers as labels. To use numeric axis, you can set the [valueType](#) property of the axis to **double**.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  primaryYAxis: {  
    //Use numerical scale in primary Y axis  
    valueType: 'double',  
    // ...  
  },  
  // ...  
});
```



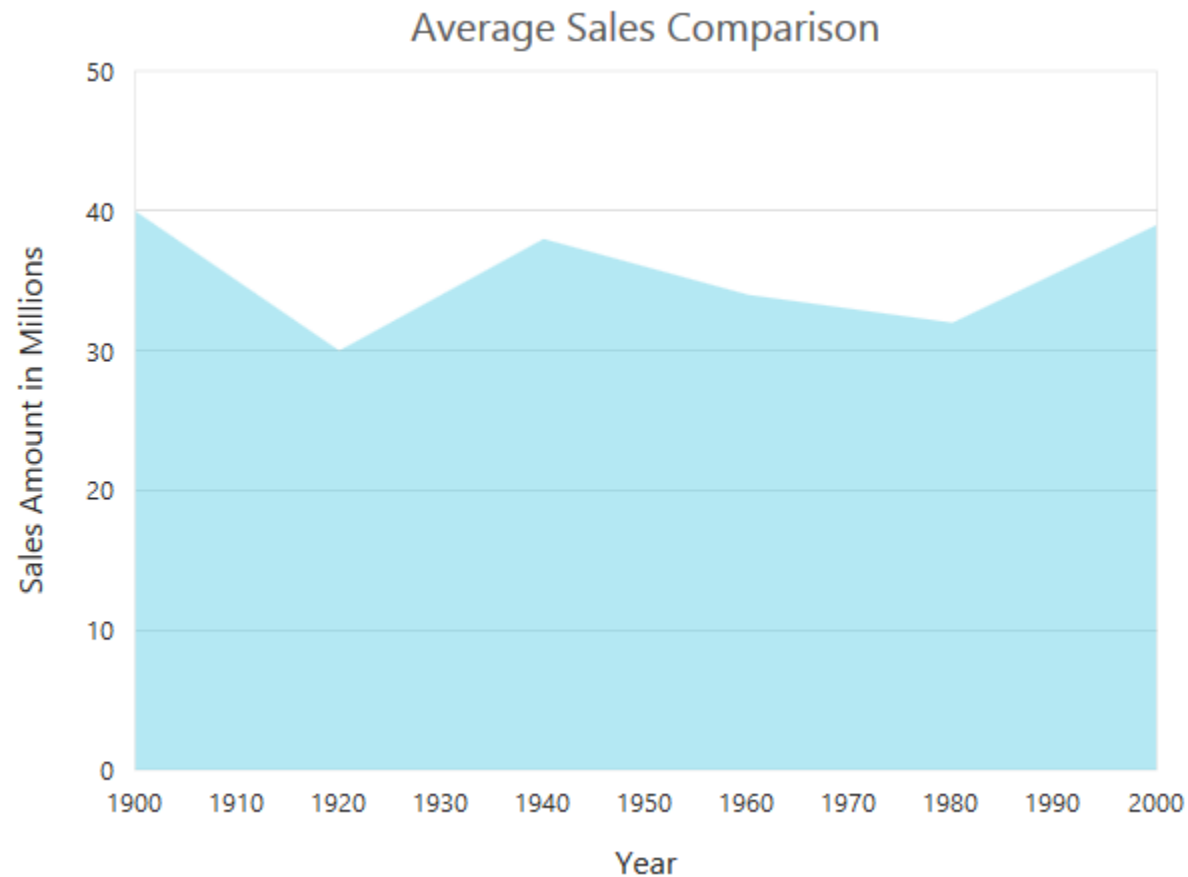
#### Customize numeric range

To customize the range of an axis, you can use the [range](#) property of the axis to set the [minimum](#), [maximum](#) and [interval](#) values. Nice range is calculated automatically based on the provided data, by default.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
});
```

```
primaryYAxis: {  
  //Customizing Y-axis range  
  range: { min: 0, max: 50 },  
  // ...  
},  
// ...  
});
```

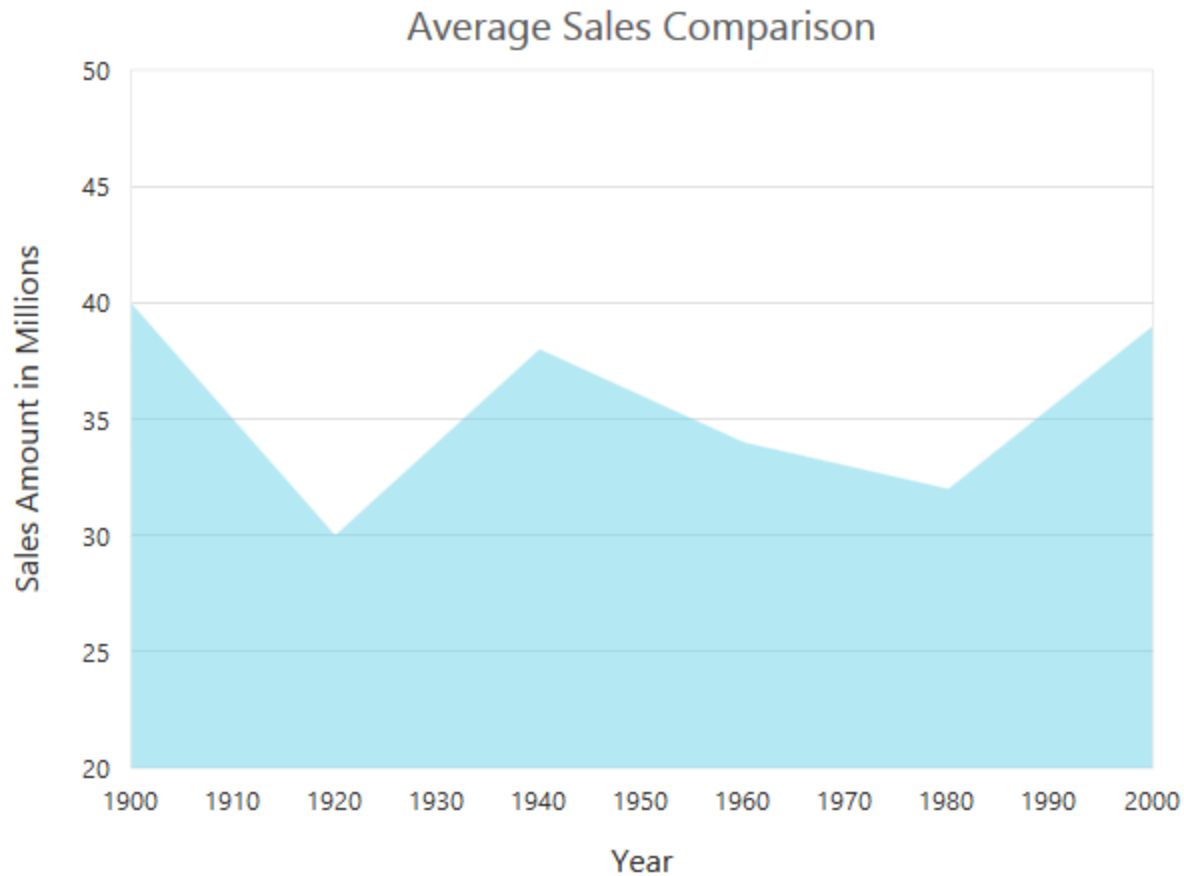


#### Customizing numeric interval

Axis interval can be customized by using the [interval](#) property of the axis range. Nice interval is calculated based on the minimum and maximum value of the provided data, by default.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  primaryYAxis: {  
    //Customizing Y-axis interval  
    range: { interval: 5 },  
    // ...  
  },  
  // ...  
});
```



#### *Apply padding to the range*

Padding can be applied to the minimum and maximum extremes of the axis range by using the [rangePadding](#) property. Numeric axis supports the following types of padding

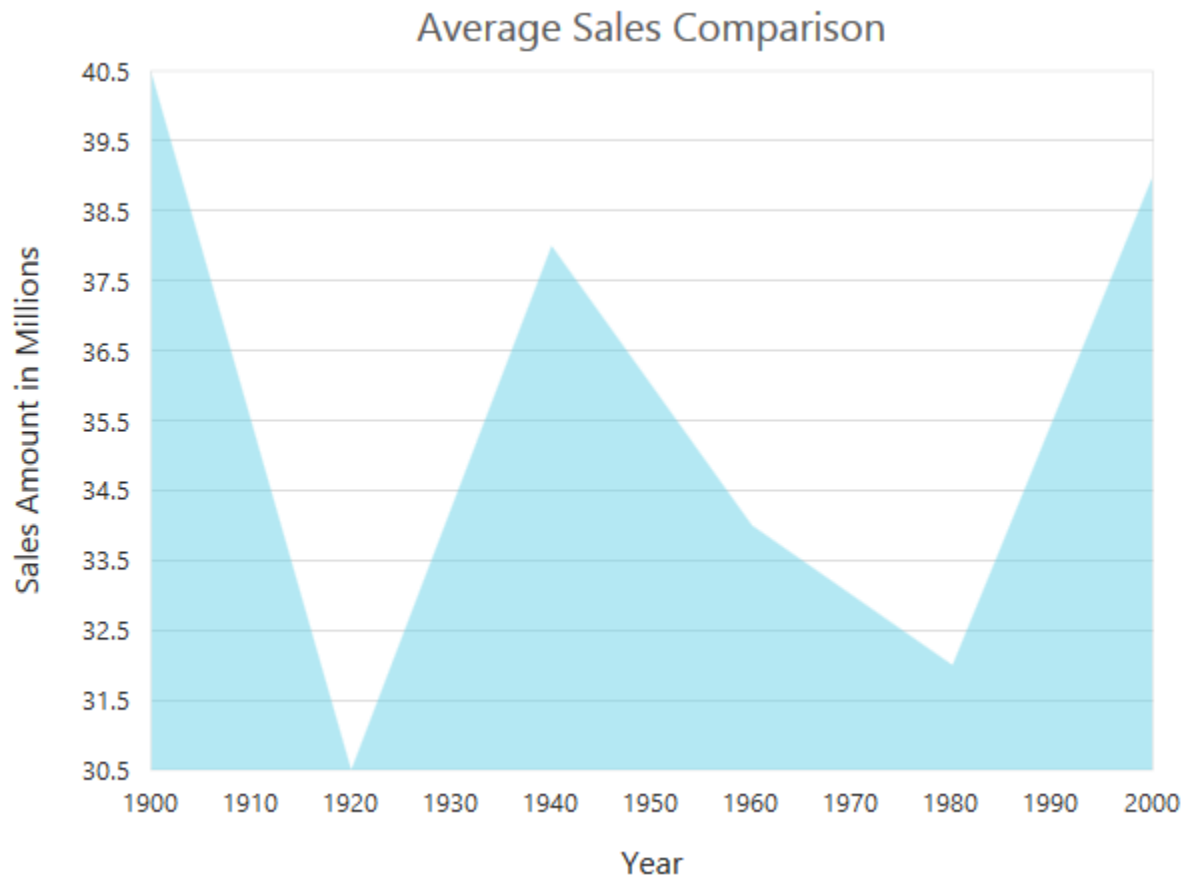
- None
- Round
- Additional
- Normal

#### **None**

When the value of the [rangePadding](#) property is **none**, padding can not be applied to the axis. This is also the default value of the rangePadding.

#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  primaryYAxis: {  
    //Applying none as range padding  
    rangePadding: 'none',  
    // ...  
  },  
  // ...  
});
```



#### Round

When the value of [rangePadding](#) property is **round**, the axis range is rounded to the nearest possible value divided by the interval.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  primaryYAxis: {  
    //Applying round as range padding  
    rangePadding: 'round',  
    // ...  
  },  
  // ...  
});
```

#### Chart before rounding axis range

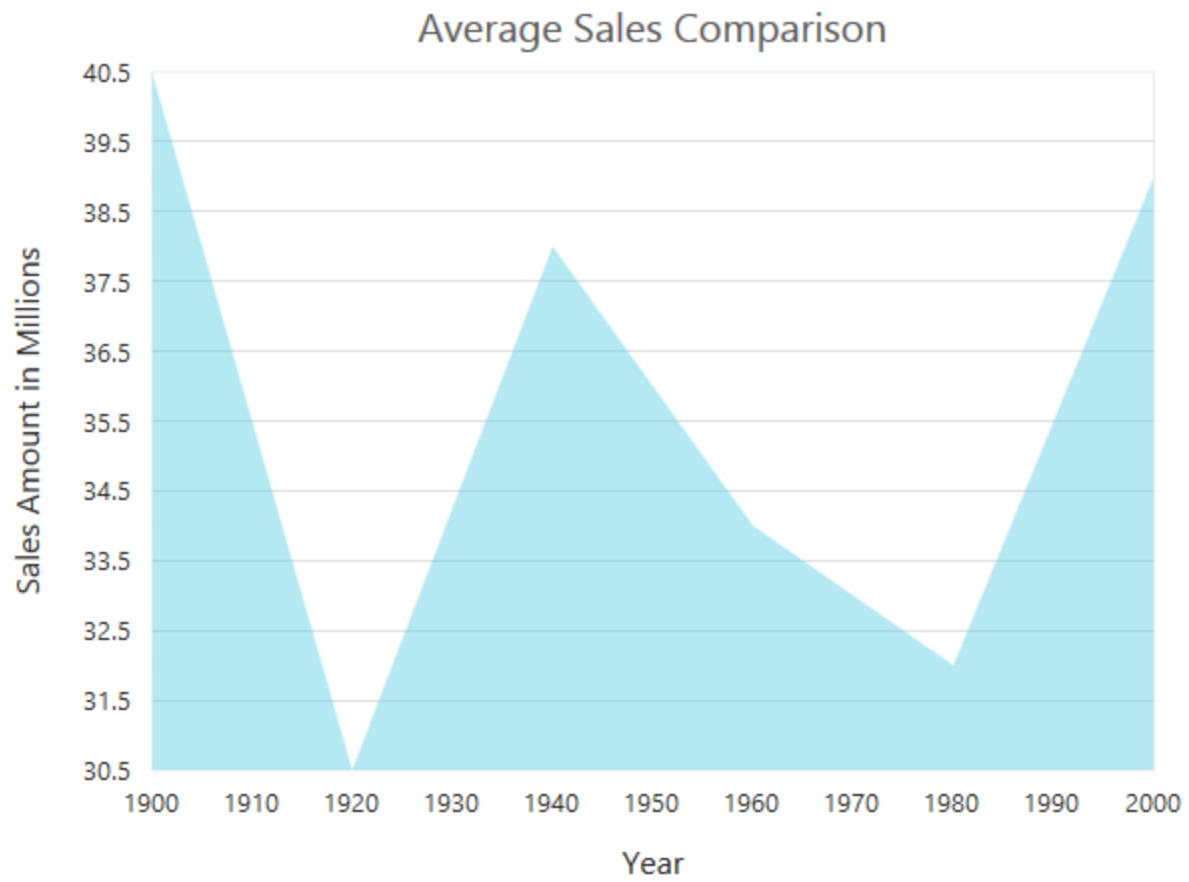
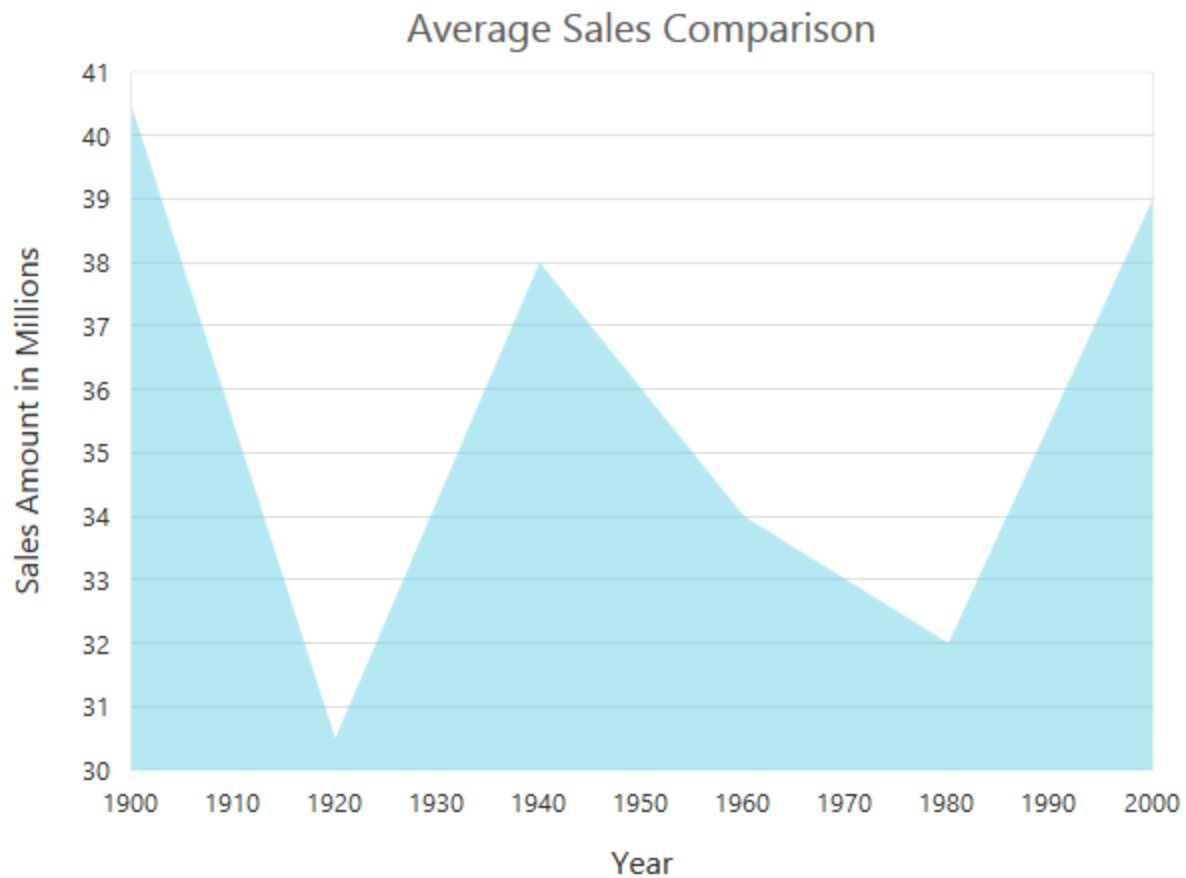


Chart after rounding axis range

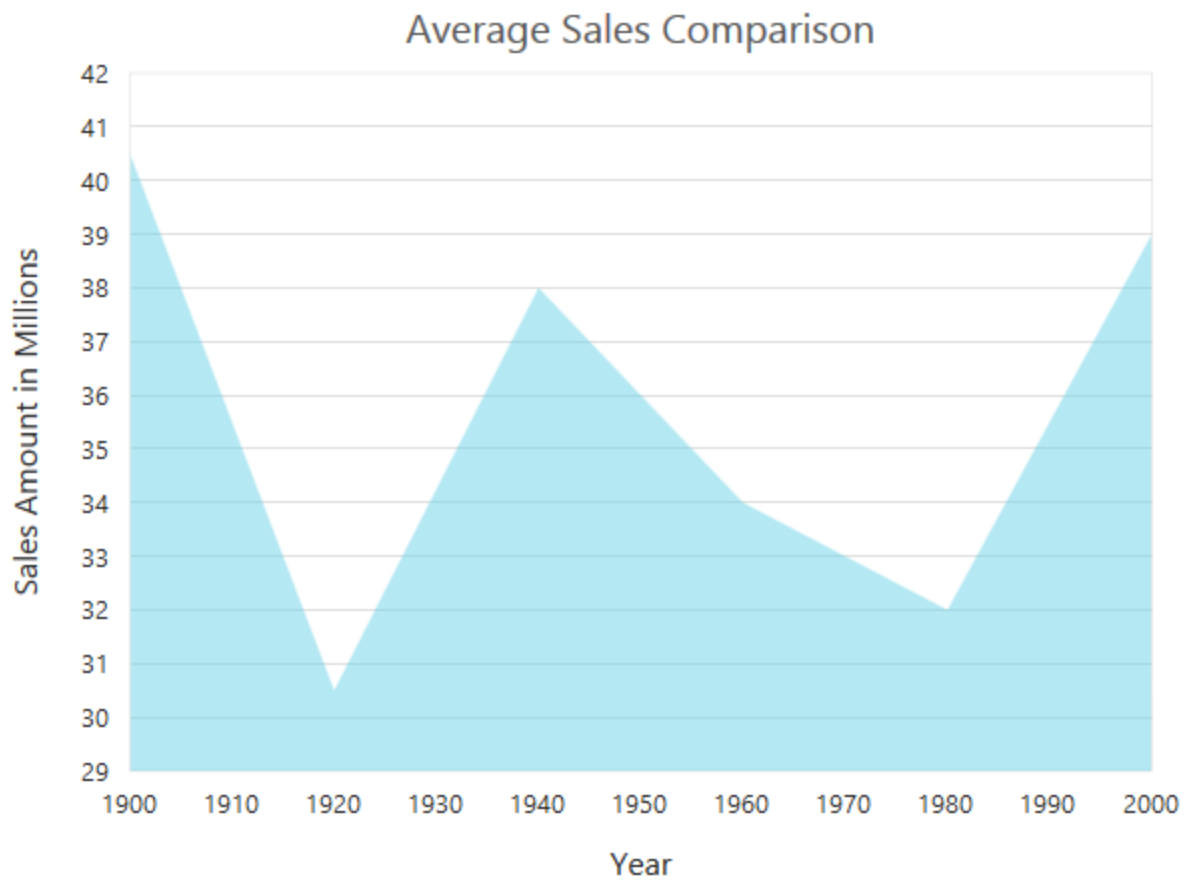


#### Additional

When the value of the [rangePadding](#) property is **additional**, the axis range is rounded and an interval of the axis is added as padding to the minimum and maximum values of the range.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  primaryYAxis: {  
    //Applying additional range padding  
    rangePadding: 'additional',  
    // ...  
  },  
  // ...  
});
```



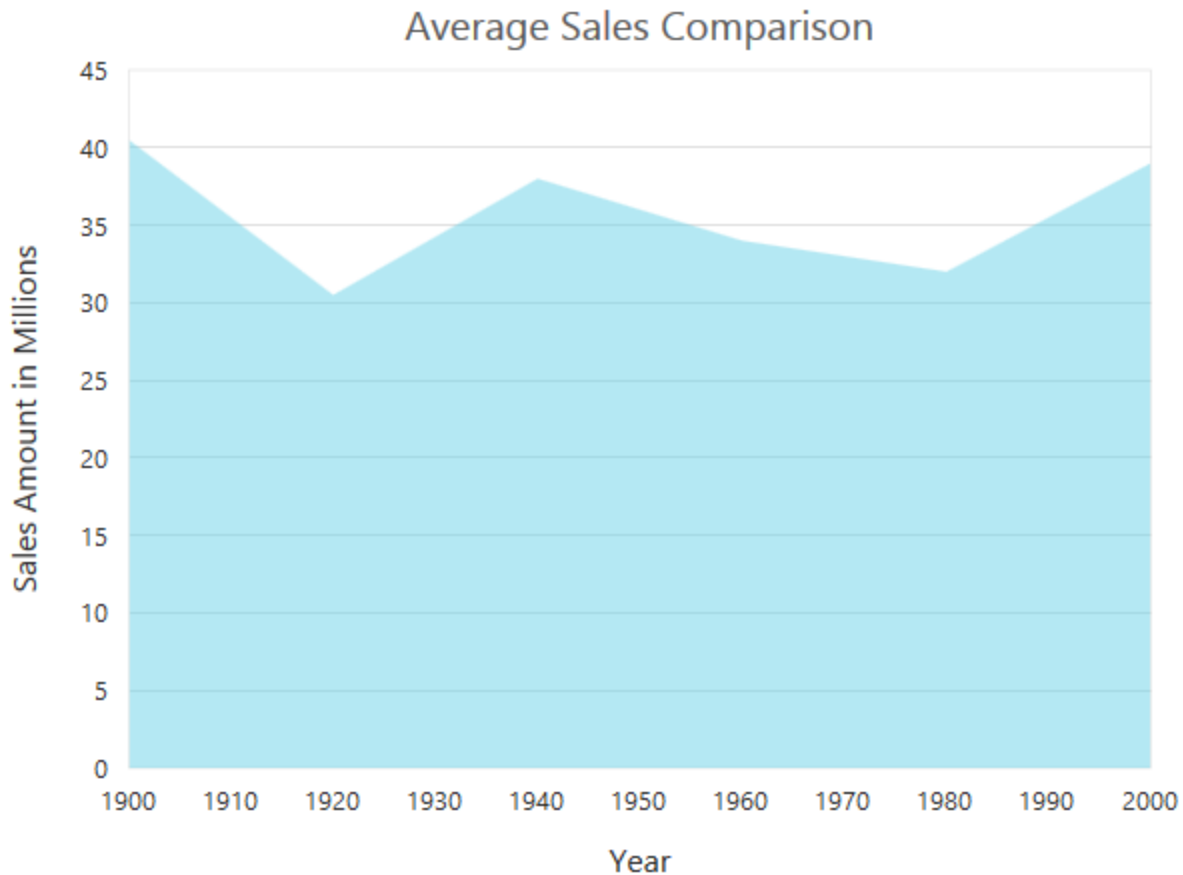
#### Normal

When the value of the [rangePadding](#) property is **normal**, the padding is applied to the axis based on the range calculation.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  primaryYAxis: {  
    //Applying normal range padding  
    rangePadding: 'normal',  
    // ...  
  },  
  // ...  
});
```



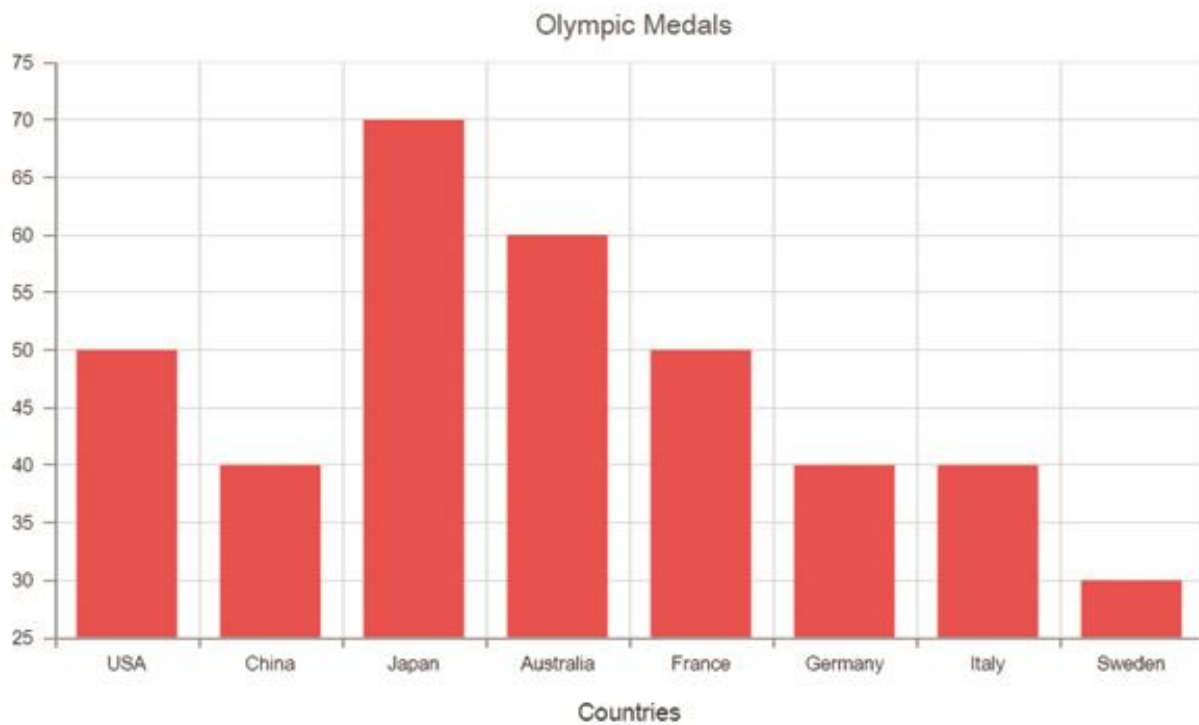


#### Customizing the starting range of the axis

By default the Y axis will be always calculated from the value 0 for column, bar, stacking column and stacking bar series types. You can modify this behavior by setting false to the property [startFromZero](#) in the axis. On setting this the axis minimum value will be calculated based on the value for the data points.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  {
    //Initializing Primary Y Axis
    primaryYAxis:
    {
      rangePadding: "none",
      startFromZero: false
    },
    // ..
  })
});
```

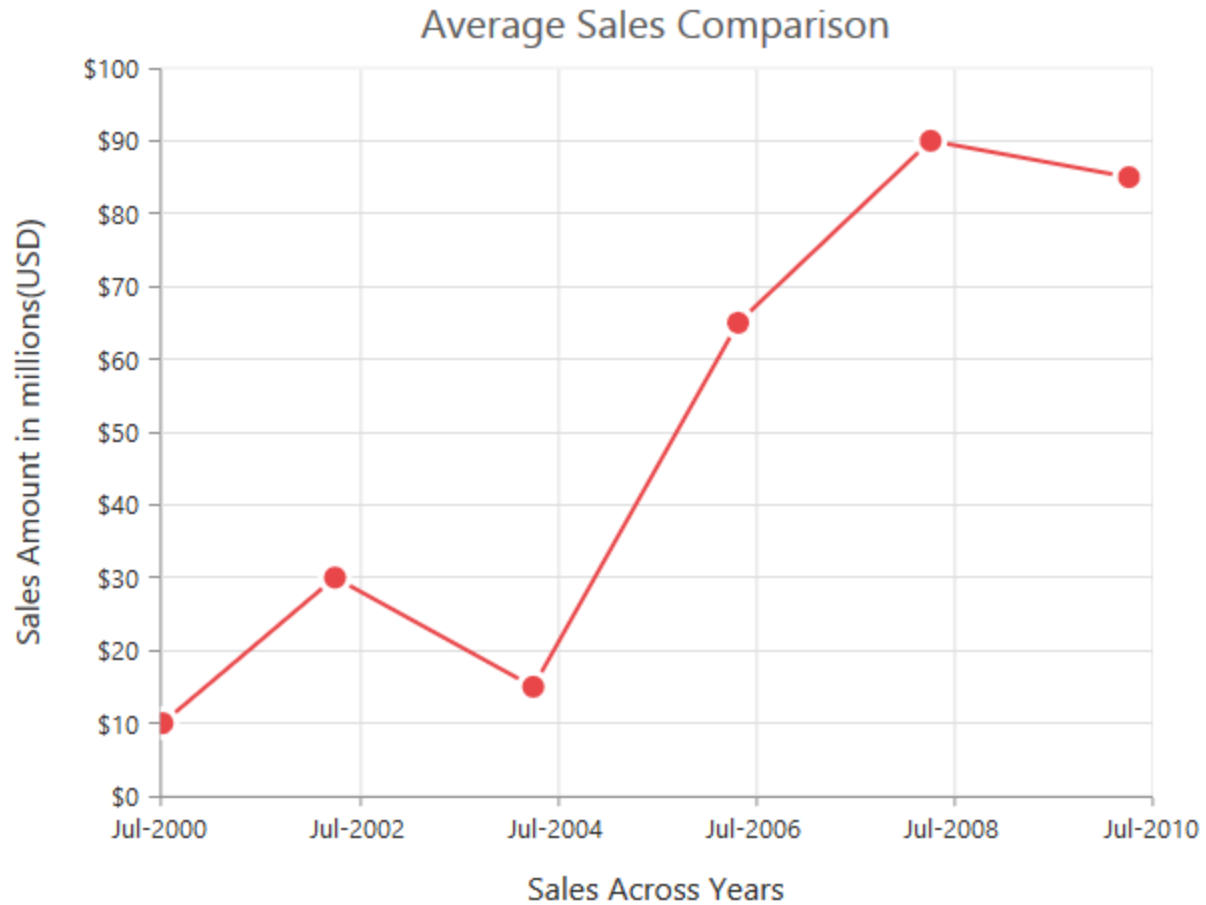


### DateTime Axis

Date time axis uses date time scale and displays the date time values as axis labels in the specified format. To use date time axis, set the [valueType](#) property of the axis to **datetime**.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  primaryXAxis: {  
    //Use date time scale in primary X axis  
    valueType: 'datetime',  
    // ...  
  },  
  // ...  
});
```

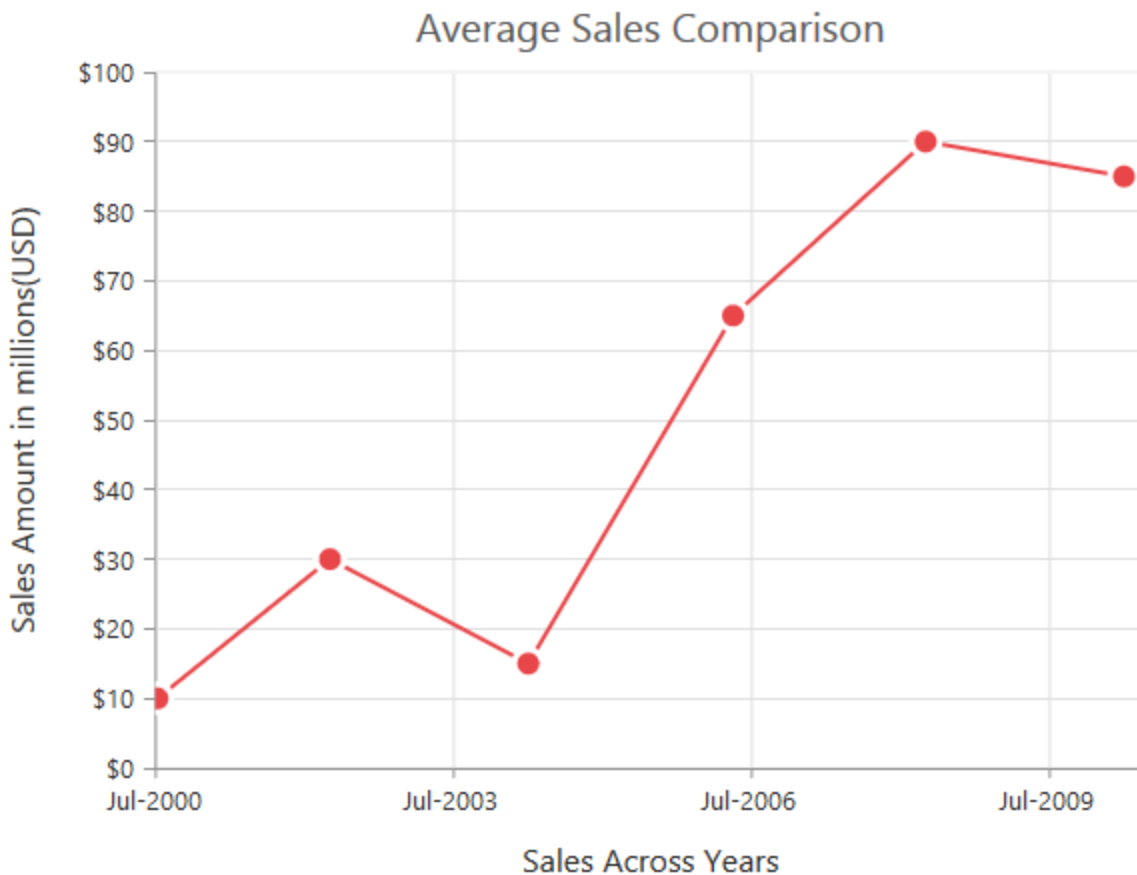


#### Customizing date time range

Axis range can be customized by using the [range](#) property to set the [minimum](#), [maximum](#) and [interval](#) values. Nice range is calculated automatically based on the provided data, by default.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  primaryXAxis: {  
    //Customizing X-axis date time range  
    range: {  
      min: new Date(2000, 6, 1),  
      max: new Date(2010, 6, 1)  
    },  
    // ...  
  },  
  // ...  
});
```



#### *Date time intervals*

Date time intervals can be customized by using the [interval](#) and [intervalType](#) properties of the axis. For example, when you set [interval](#) as **2** and [intervalType](#) as **years**, it considers the 2 years as interval.

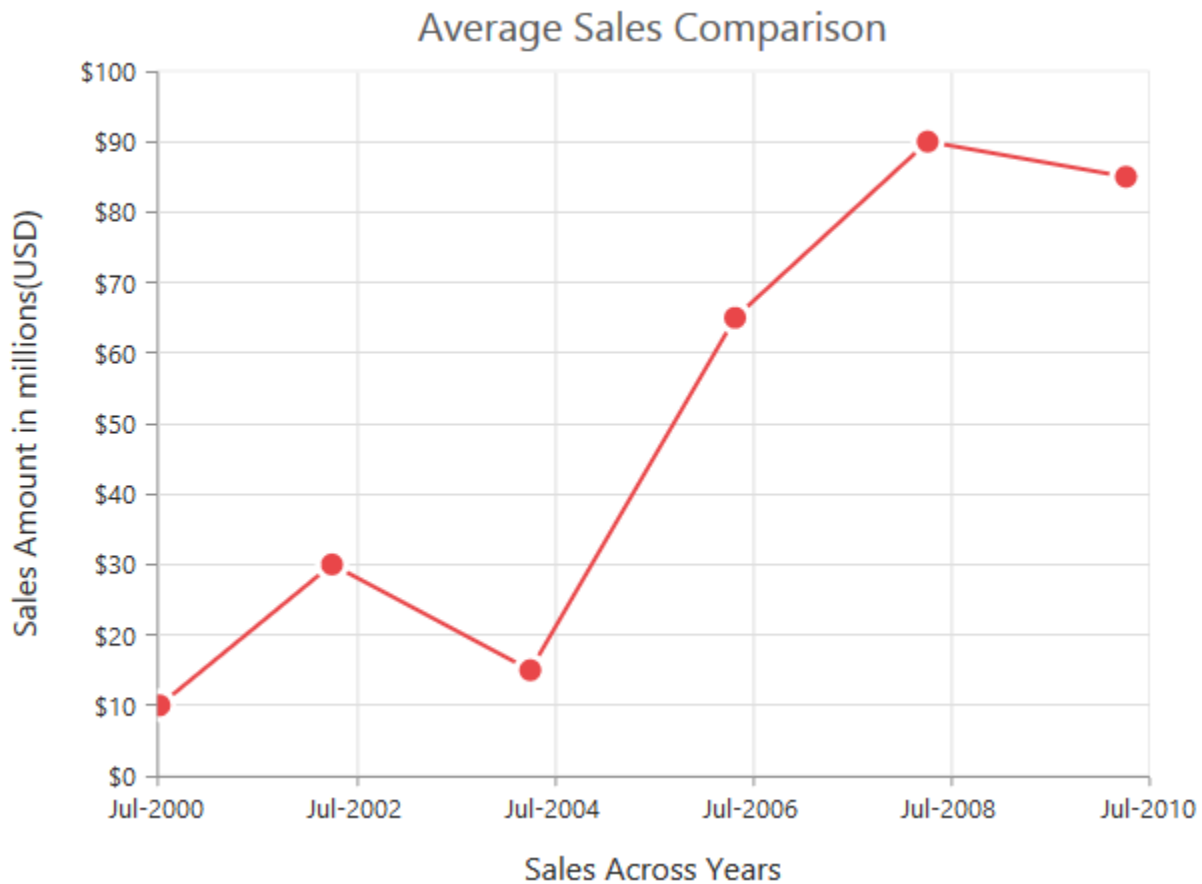
Essential Chart supports the following types of interval for date time axis.

- Days
- Hours
- Milliseconds
- Minutes
- Months
- Seconds
- Years

#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Customizing X-axis date time range
    range: {
      interval: 2,
    },
    intervalType: 'years',
    // ...
  }
});
```

```
},
// ...
});
```



#### Apply padding to the range

Padding can be applied to the minimum and maximum extremes of the range by using the [rangePadding](#) property. Date time axis supports the following types of padding

- None
- Round
- Additional

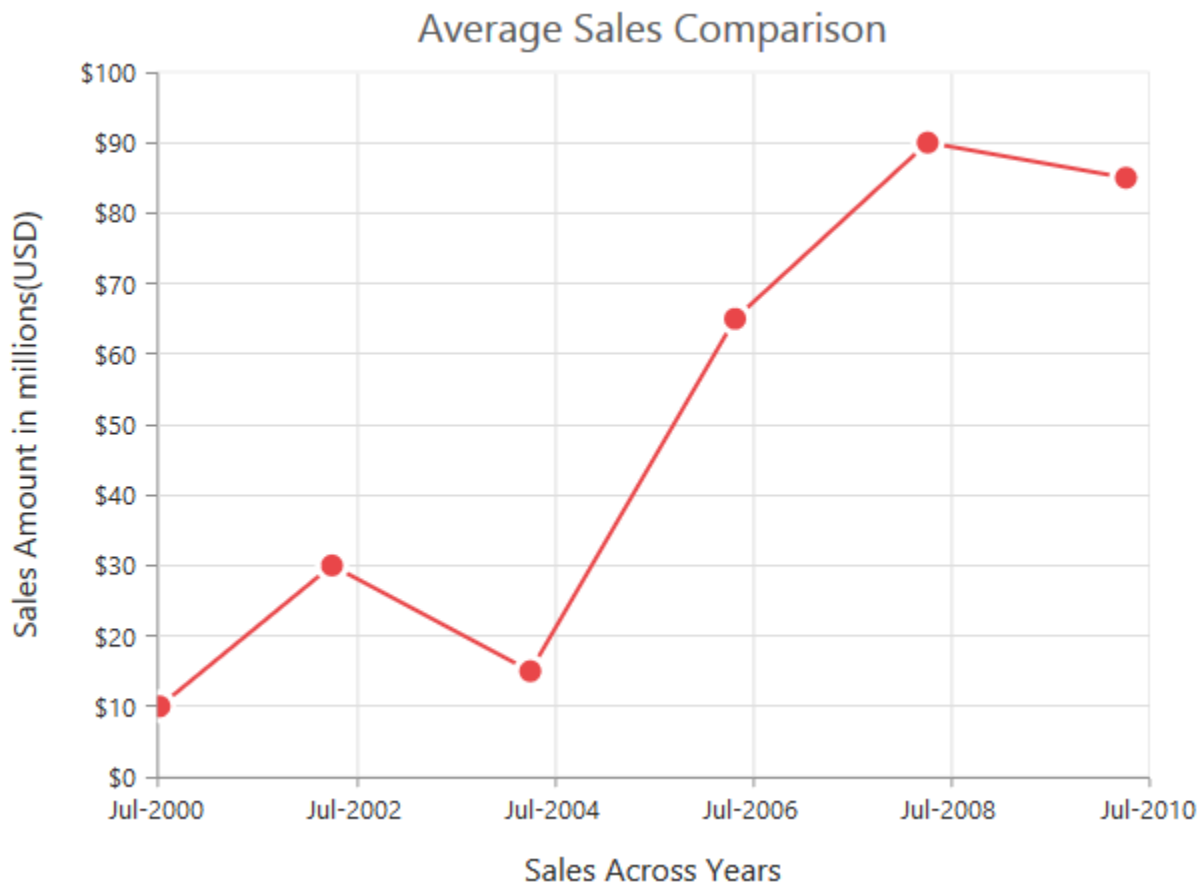
#### None

When the value of the [rangePadding](#) property is **none**, padding is applied to the axis. This is also the default value of the rangePadding.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Applying none as range padding
    rangePadding: 'none',
    // ...
  },
});
```

```
// ...
});
```



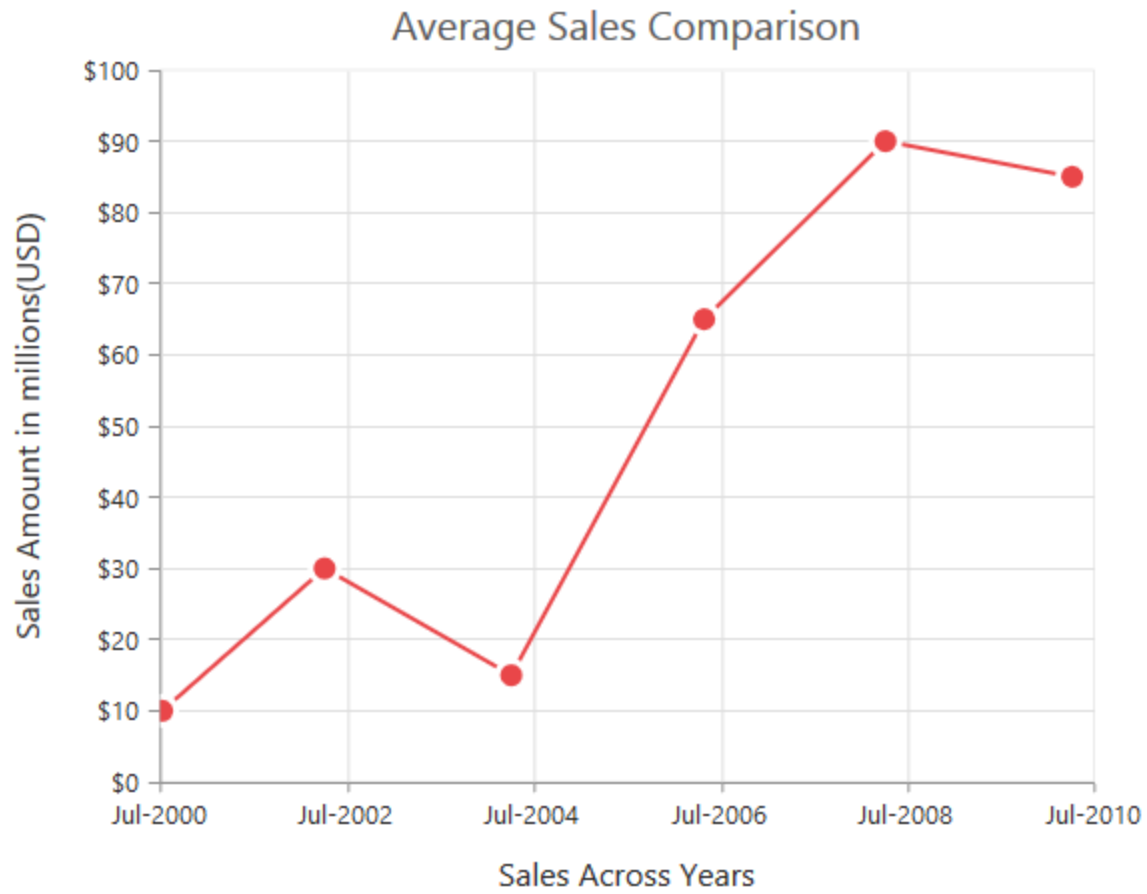
### Round

When the value of the [rangePadding](#) property is **round**, the axis range is rounded to the nearest possible date time value.

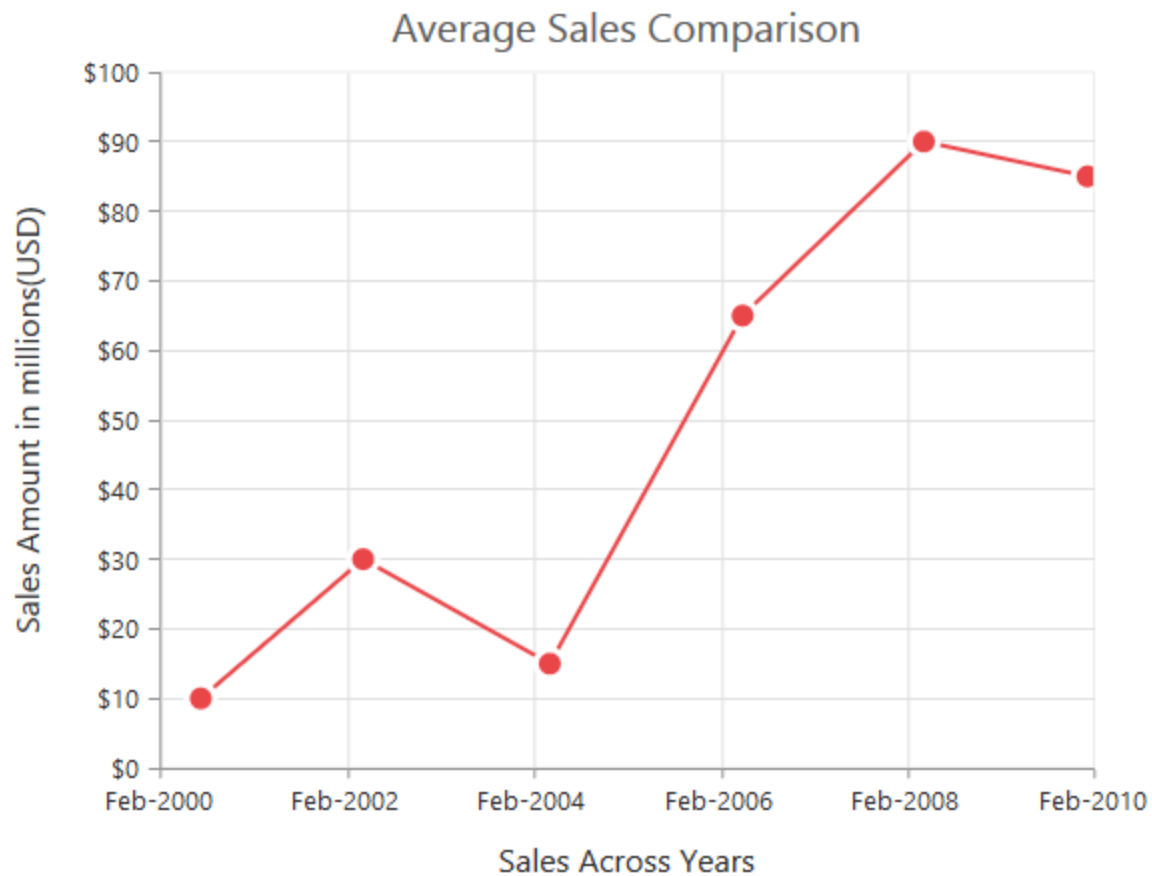
### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Applying round as range padding
    rangePadding: 'round',
    // ...
  },
  // ...
});
```

### Chart before rounding axis range



**Chart after rounding axis range**



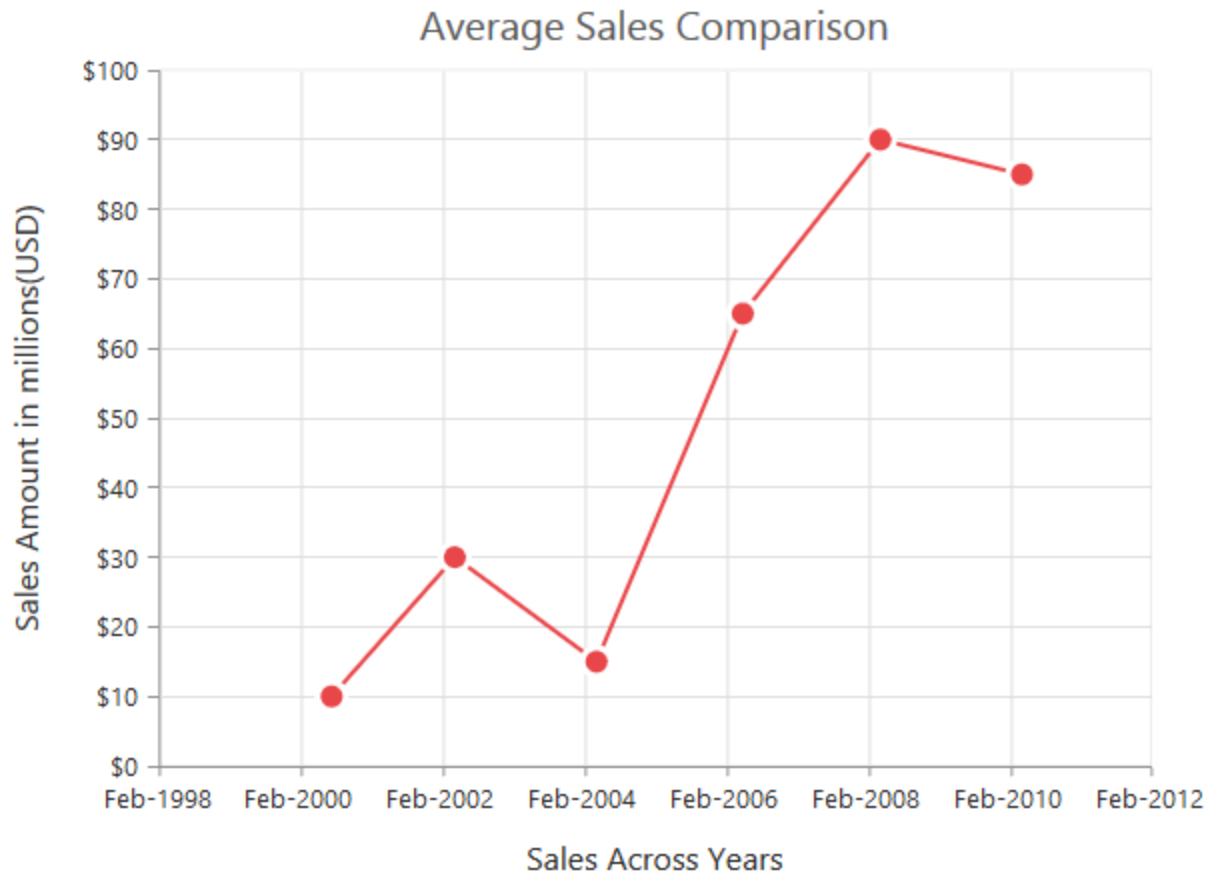
### Additional

When the value of the [rangePadding](#) property is **additional**, the range is rounded and date time interval of the axis are added as padding to the minimum and maximum extremes of the range.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Applying additional as range padding
    rangePadding: 'additional',
    // ...
  },
  // ...
});
```



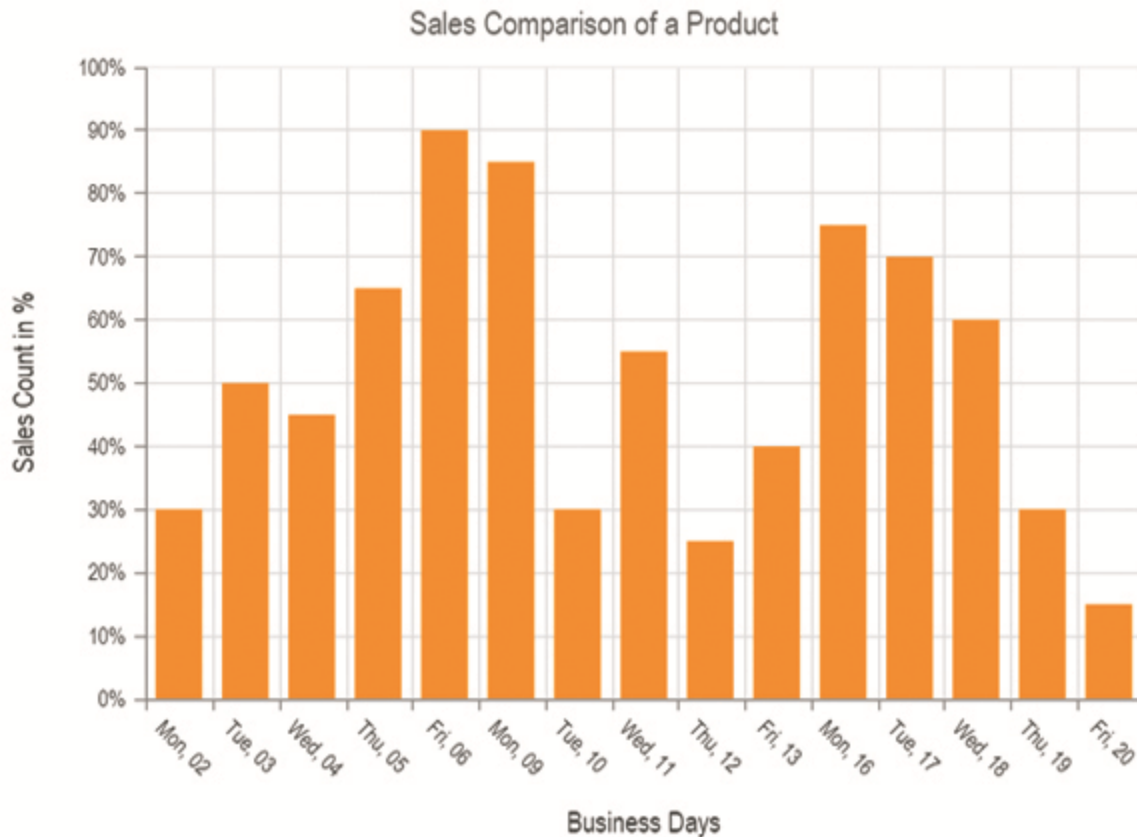


### DateTime Category Axis

DateTime category axis takes date time value as input but behaves like category axis. This is used to display the date time values with nonlinear intervals (used to depict the business days by skipping holidays). To use date time axis, set the [valueType](#) property of the axis to **datetimeCategory**.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    valueType: 'datetimeCategory',
    // ...
  },
  // ...
});
```



#### Customizing DateTime Category range

Axis range can be customized by using the [range](#) property to set the [minimum](#), [maximum](#) and [interval](#) values. Datetime category axis takes numeric input for minimum and maximum property.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    range: { //Customizing X-axis date time category range
      min: 0,
      max: 4
    },
    // ...
  },
  // ...
});
```



#### *Date/Time Category intervals*

Date time category intervals can be customized by using the [interval](#) and [intervalType](#) properties of the axis. For example, when you set the `intervalType` as months, it displays only the first label of all the months from the data.

Essential Chart supports the following types of interval for date time category axis.

- Days
- Hours
- Milliseconds
- Minutes
- Months
- Seconds
- Years
- Auto

#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    intervalType: "months"
    // ...
  },
  // ...
});
```

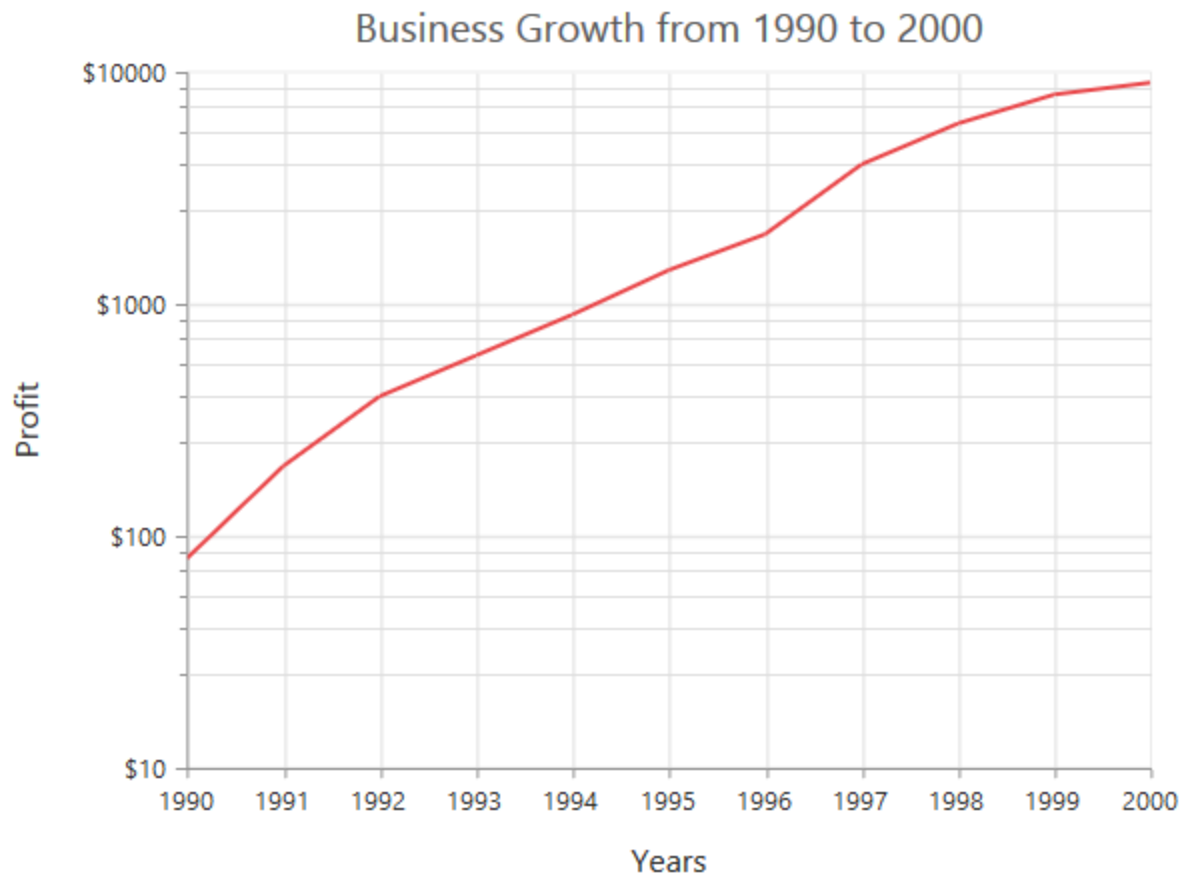


### Logarithmic Axis

Logarithmic axis uses logarithmic scale and it is very useful in visualizing when the data has values with both lower order of magnitude (eg:  $10^{-6}$ ) and higher order of magnitude (eg:  $10^6$ ). To use logarithmic axis, set the [valueType](#) property of the axis to **logarithmic**.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Use logarithmic scale in primary X axis
    valueType: 'logarithmic',
    // ...
  },
  // ...
});
```

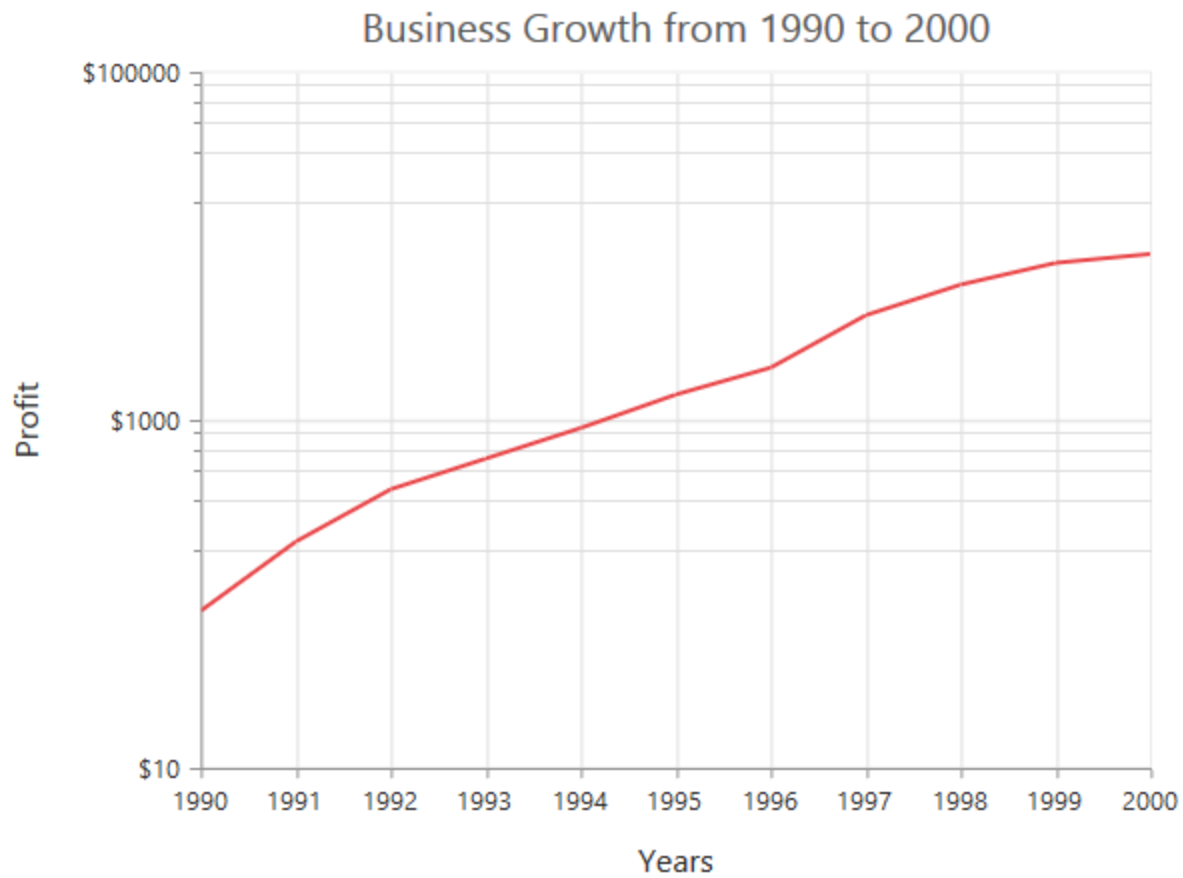


#### Customize Logarithmic range

Logarithmic range can be customized by using the [range](#) property of the axis to change the [minimum](#), [maximum](#) and [interval](#) values. Nice range is calculated automatically based on the provided data, by default.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryYAxis: {
    //Customizing logarithmic range
    range: { min: 1, max: 5 },
    // ...
  },
  // ...
});
```

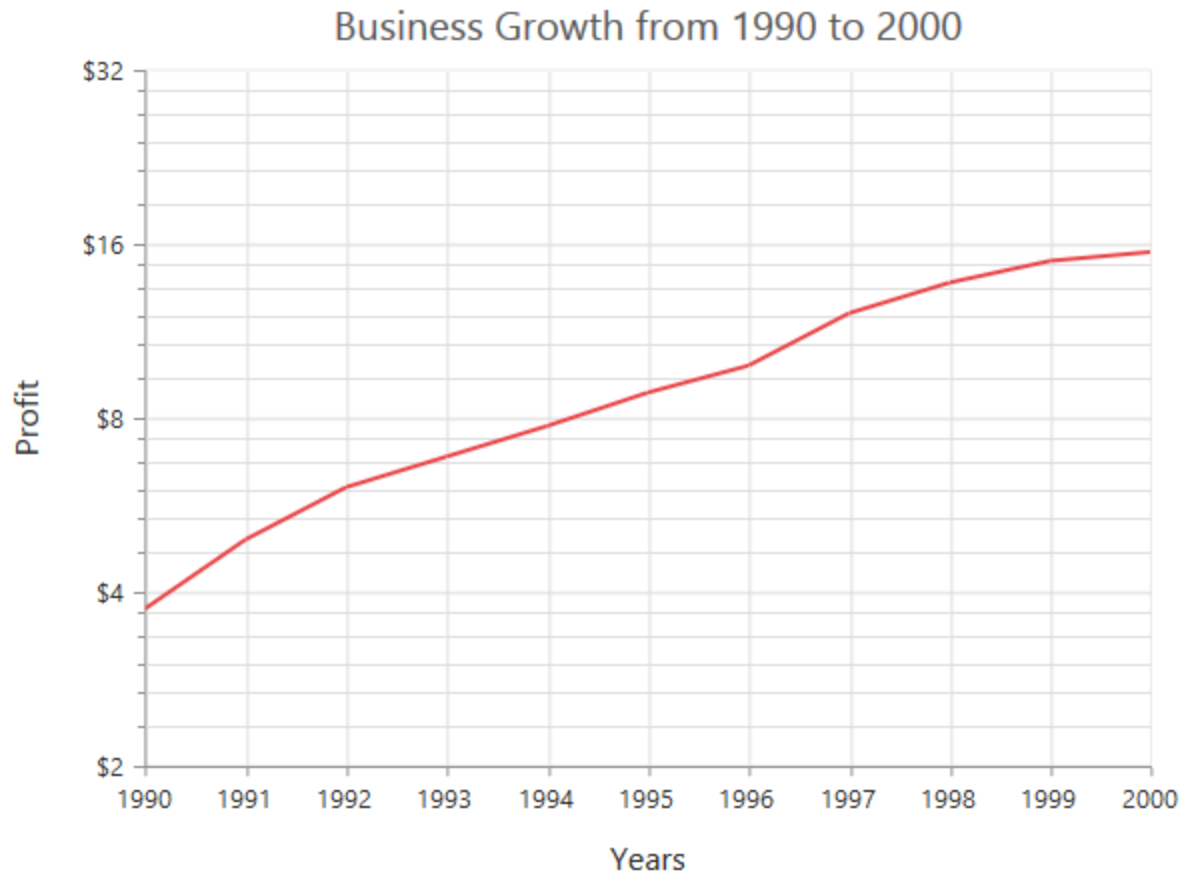


#### Logarithmic base

Logarithmic base can be customized by using the [logBase](#) property of the axis. The default value of the [logBase](#) is 10.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryYAxis: {
    //Customizing logarithmic base
    logBase: 2,
    // ...
  },
  // ...
});
```

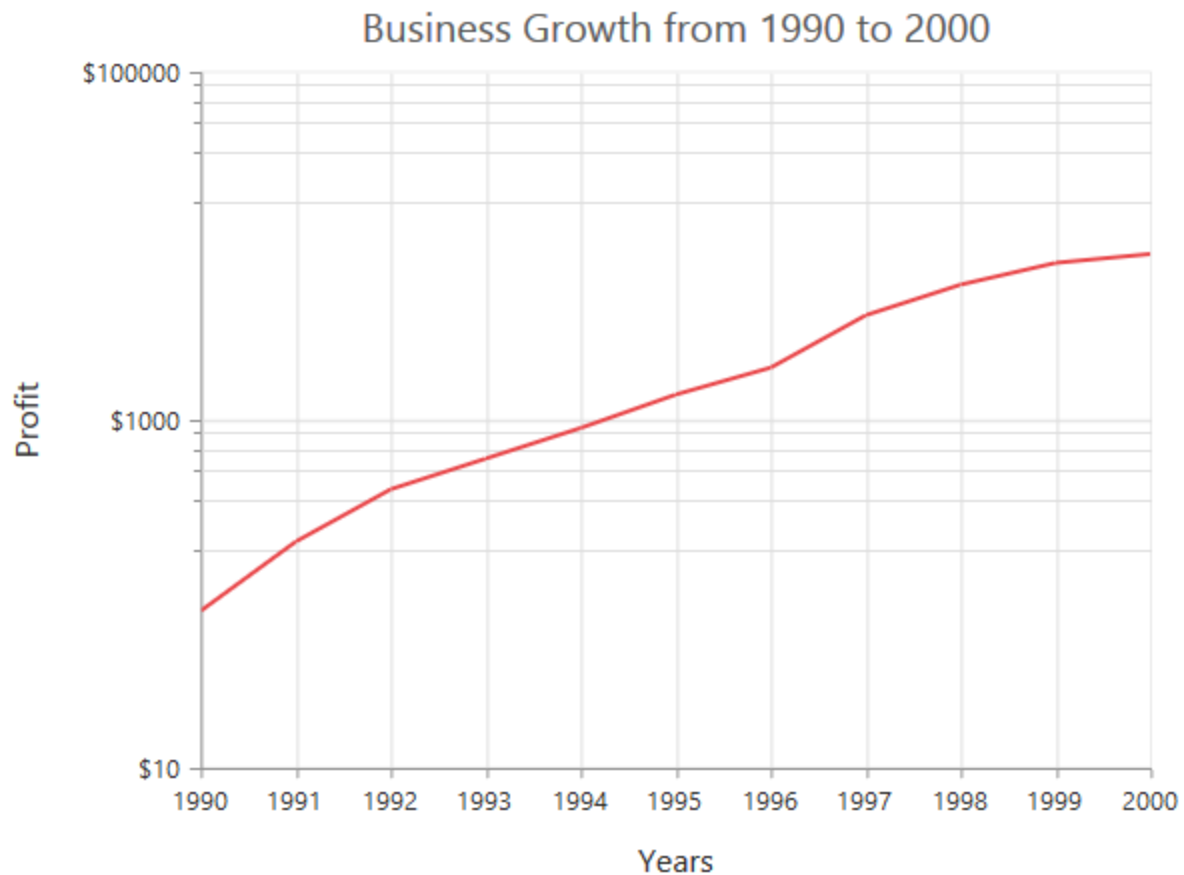


#### Logarithmic interval

Logarithmic axis interval can be customized by using the [interval](#) property of the axis. When the logarithmic base is 10 and logarithmic interval is 2, then the axis labels are placed at an interval of  $10^2$ . The default value of the interval is 1.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryYAxis: {
    //Customizing logarithmic interval
    range: { interval: 2 },
    // ...
  },
  // ...
});
```



#### Label Format

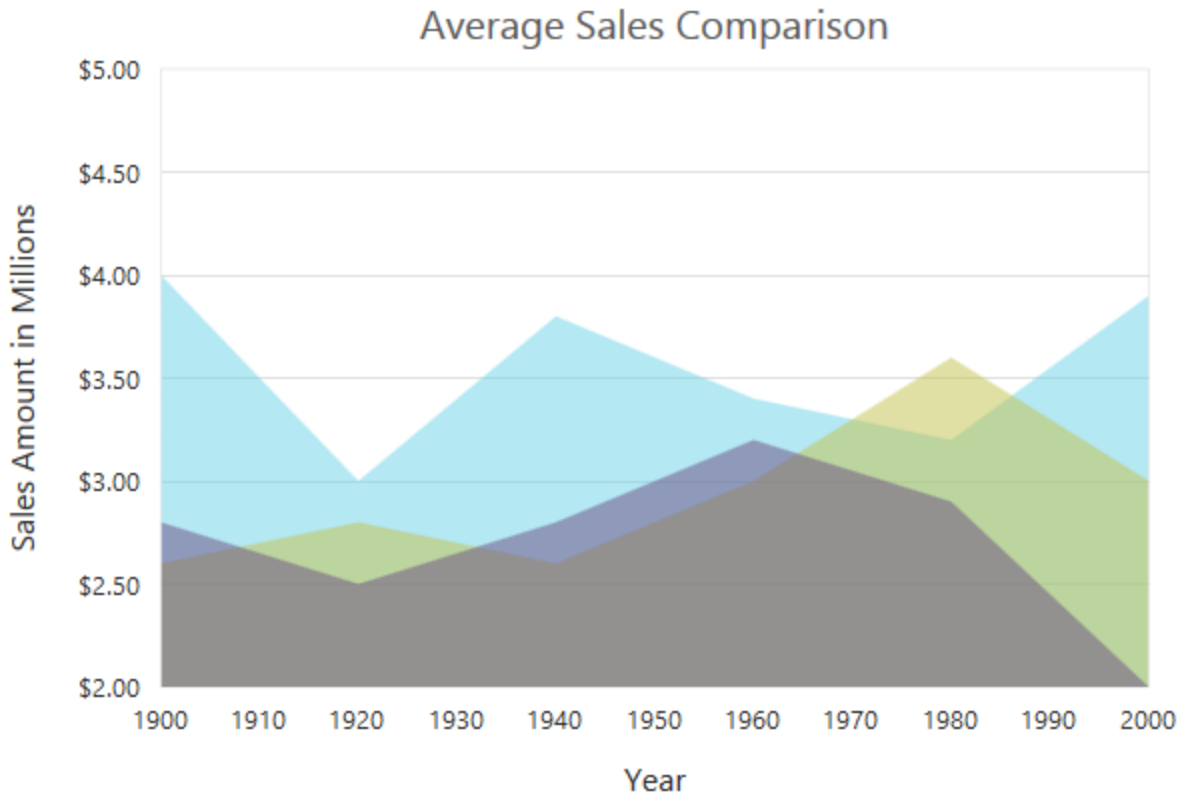
##### *Format numeric labels*

Numeric labels can be formatted by using the [labelFormat](#) property. Numeric values can be formatted with n (number with decimal points), c (currency) and p (percentage) commands.

#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Applying currency format to axis labels
    labelFormat: 'c',
    // ...
  },
  // ...
});
```





The following table describes the result of applying some commonly used label formats on numeric values.

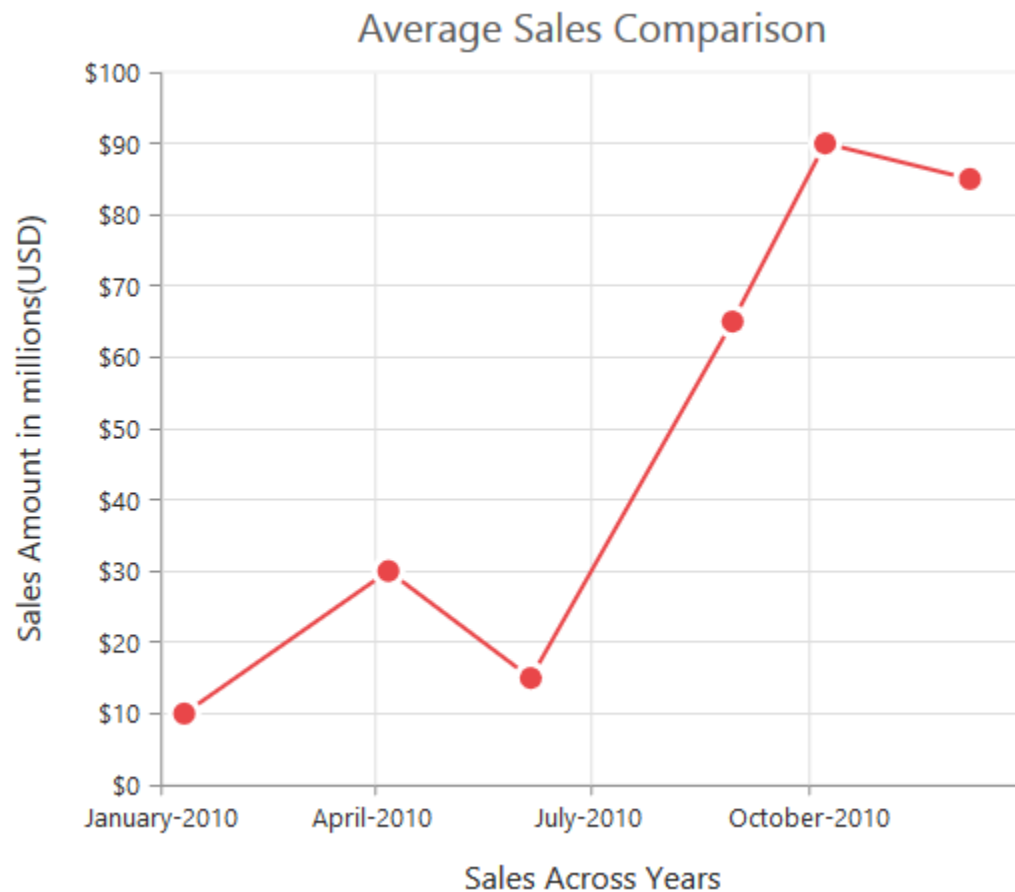
| Label Value | Label Format property value | Result     | Description  |
|-------------|-----------------------------|------------|--|
| 1000        | n1                          | 1000.0     | The Number is rounded to 1 decimal place   |
| 1000        | n2                          | 1000.00    | The Number is rounded to 2 decimal place   |
| 1000        | n3                          | 1000.000   | The Number is rounded to 3 decimal place   |
| 0.01        | p1                          | 1.0%       | The Number is converted to percentage with 1 decimal place                         |
| 0.01        | p2                          | 1.00%      | The Number is converted to percentage with 2 decimal place                         |
| 0.01        | p3                          | 1.000%     | The Number is converted to percentage with 3 decimal place                         |
| 1000        | c1                          | \$1,000.0  | The Currency symbol is appended to number and number is rounded to 1 decimal place |
| 1000        | c2                          | \$1,000.00 | The Currency symbol is appended to number and number is rounded to 2 decimal place |

### Format date time values

Date time labels can be formatted by using the [labelFormat](#) property of the axis.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Formatting date time labels in date/Month name/Year format
    labelFormat: 'dd/MMMM/yyyy',
    // ...
  },
  // ...
});
```



The following table describes the result of applying some common date time formats to the `labelFormat` property

| Label Value                         | Label Format Property Value | Result     | Description                                     |
|-------------------------------------|-----------------------------|------------|---|
| <code>new Date(2000, 03, 10)</code> | <code>dddd</code>           | Monday     | The Date is displayed in day format             |
| <code>new Date(2000, 03, 10)</code> | <code>MM/dd/yyyy</code>     | 04/10/2000 | The Date is displayed in month/date/year format |

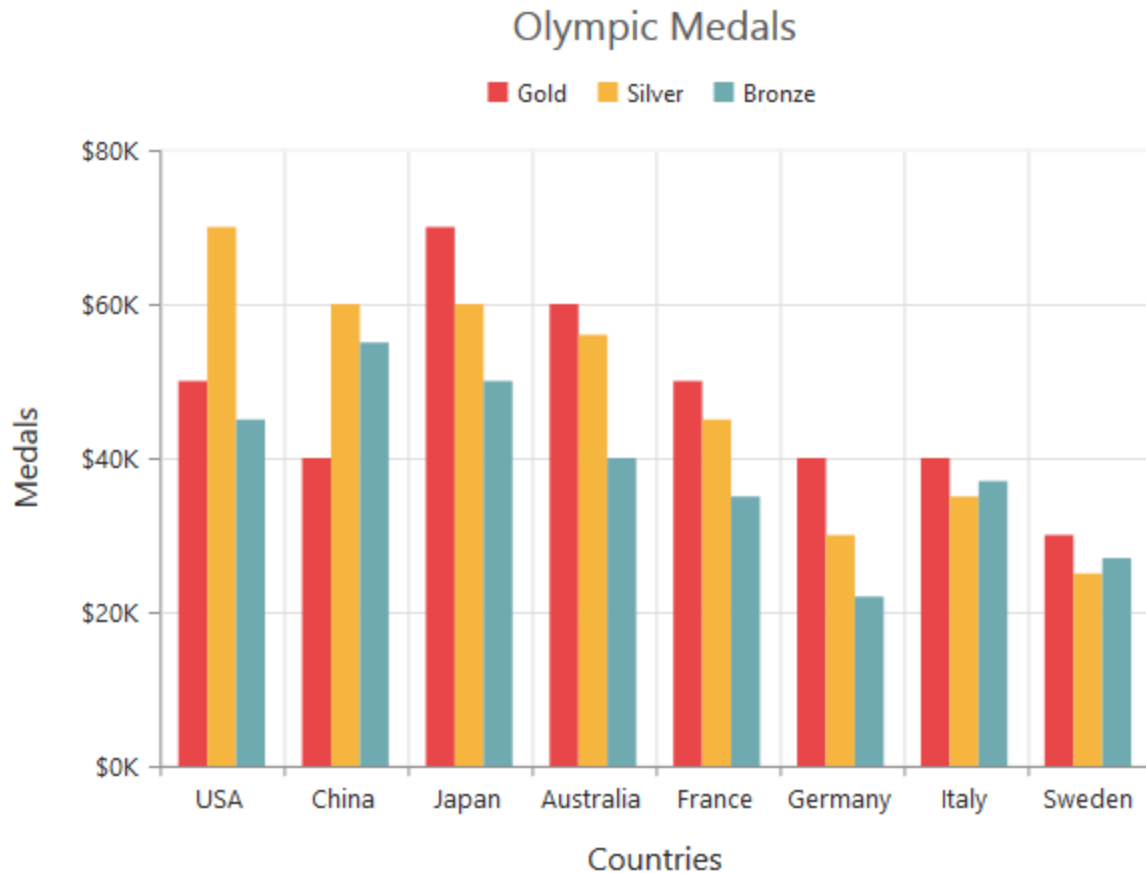
|                        |          |          |  |
|------------------------|----------|----------|--|
| new Date(2000, 03, 10) | n3       | 1000.000 | The Number is rounded to 3 decimal place               |
| new Date(2000, 03, 10) | MMM      | Apr      | The Shorthand month for the date is displayed          |
| new Date(2000, 03, 10) | t        | 12:00 AM | Time of the date value is displayed as label           |
| new Date(2000, 03, 10) | hh:mm:ss | 12:00:00 | The Label is displayed in hours:minutes:seconds format |

### Custom label format

Prefix and suffix can be added to the category labels by using the [labelFormat](#) property. You can use the `{value}` as placeholder text in your custom text, it is replaced with the corresponding axis label at the runtime.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //...
    //Adding prefix and suffix to axis labels
    labelFormat: '${value} K',
  },
  //...
});
```



#### Common axis features

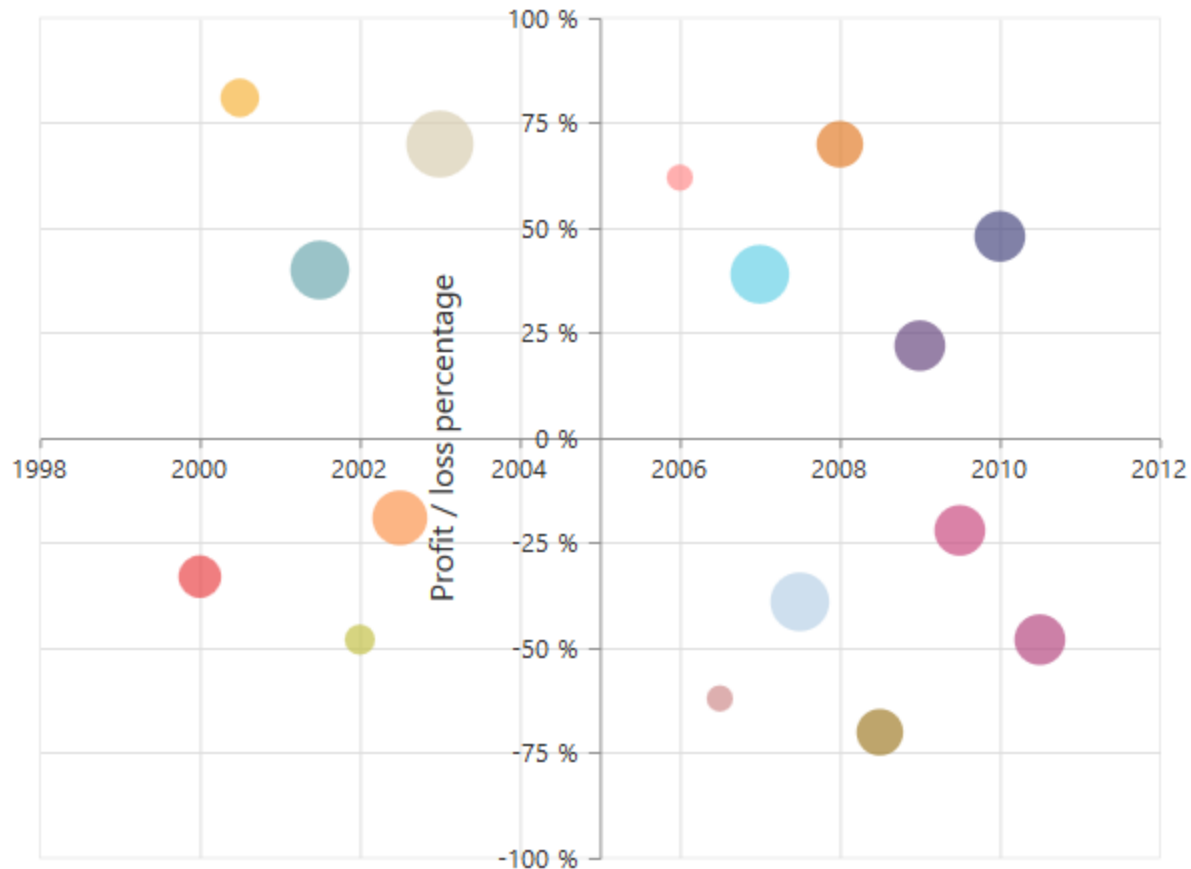
Customization of features such as axis crossing, title, labels, grid lines and tick lines are common to all the axis. Each of these features are explained in this section.

#### Axis Crossing

Axis can be positioned anywhere in chart area using the [crossesAt](#) property of axis. This property specifies where the horizontal axis should intersect or cross the vertical axis and vice versa. Default value of [crossesAt](#) property is null.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis:
  {
    //Crosses primary Y axis at 0
    crossesAt: 0,
    //...
  },
});
```

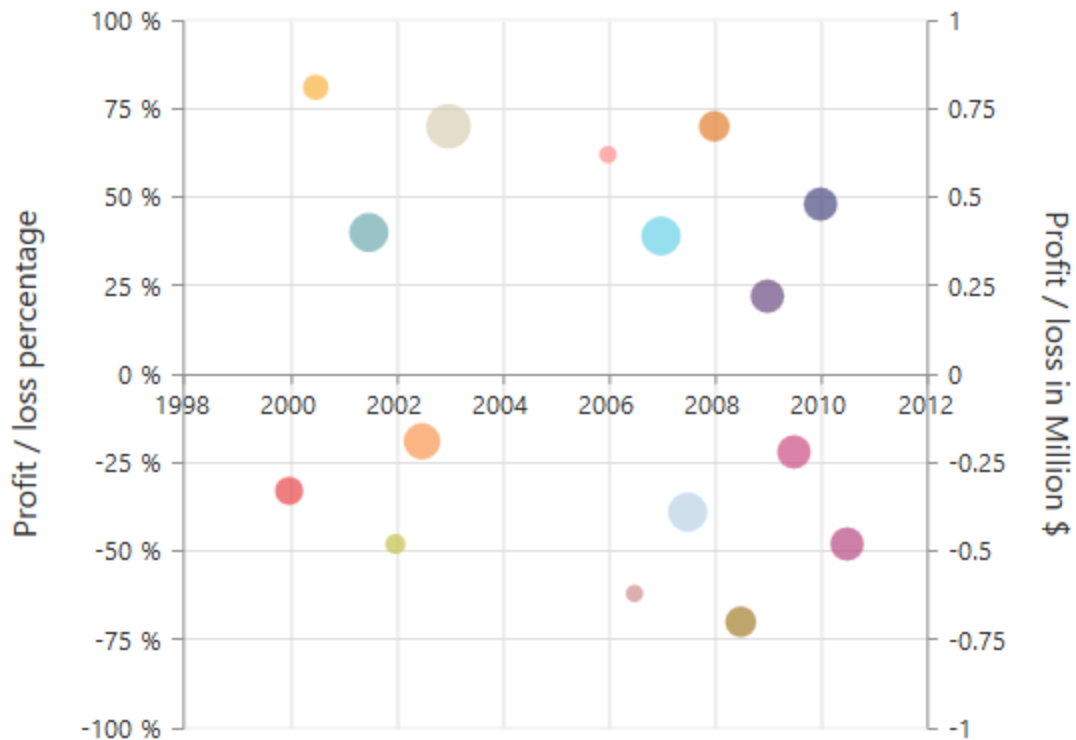


### Crossing a specific Axis

The [crossesInAxis](#) property takes axis name as input and determines the axis used for crossing. By default all the horizontal axes crosses in primary Y axis and all the vertical axes crosses in primary X axis.

### JAVASCRIPT

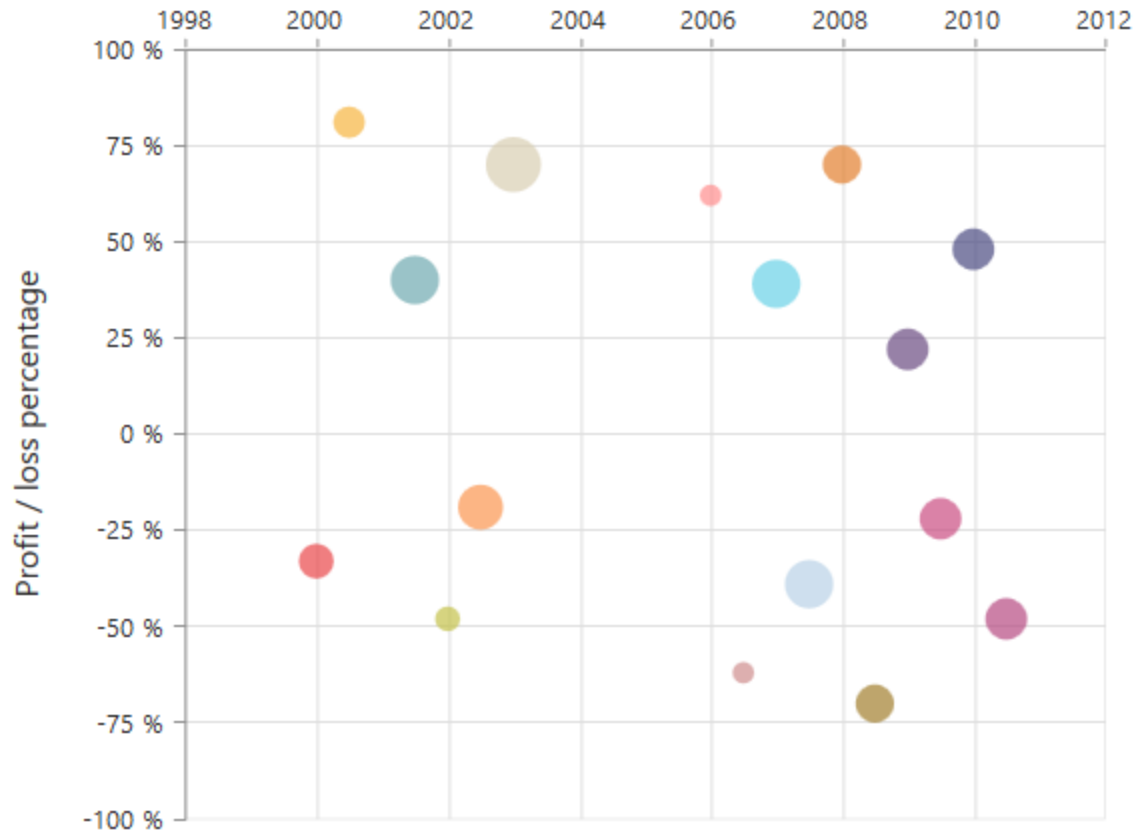
```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis:
  {
    //Crosses vertical axis at -0.2
    crossesAt: -0.2,
    //Crosses in secondary Y axis
    crossesInAxis: 'secondaryYAxis',
    //...
  },
  //Vertical axis for crossing
  axes: [{
    orientation: 'vertical',
    name: 'secondaryYAxis',
    //...
  }],
});
```



Axis will be placed in the opposite side if value of [crossesAt](#) property is greater than the maximum value of crossing axis (axis name provided through [crossesInAxis](#) property or primary Y axis for horizontal axis).

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis:
  {
    //Crosses primary Y axis at 200
    crossesAt: 200,
    //...
  },
});
```

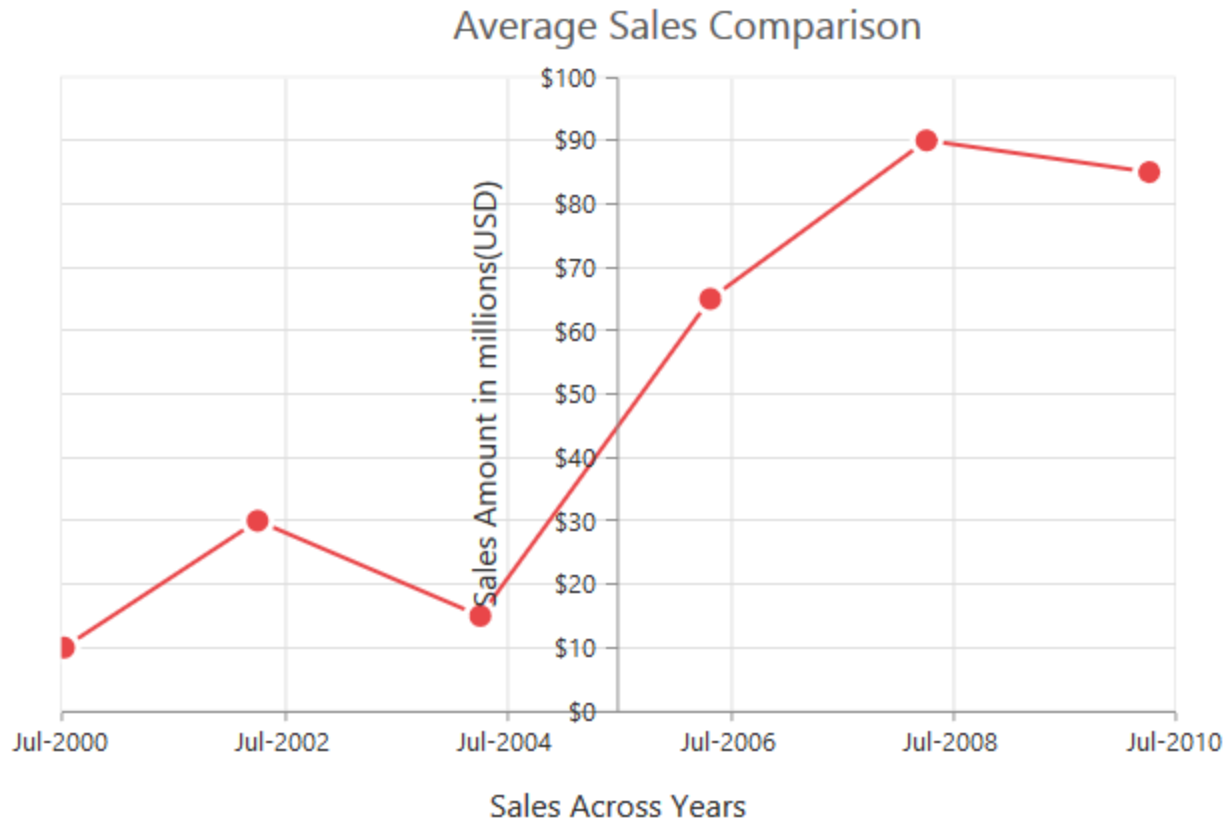


#### Crossing in DateTime Axis

For crossing in a date time horizontal axis, date object should be provided as value for [crossesAt](#) property of vertical axes.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryYAxis:
  {
    //Crosses horizontal axis at 5/29/2010
    crossesAt: new Date(2010, 4, 29),
    //...
  }
  //...
});
```



#### Crossing in Category Axis

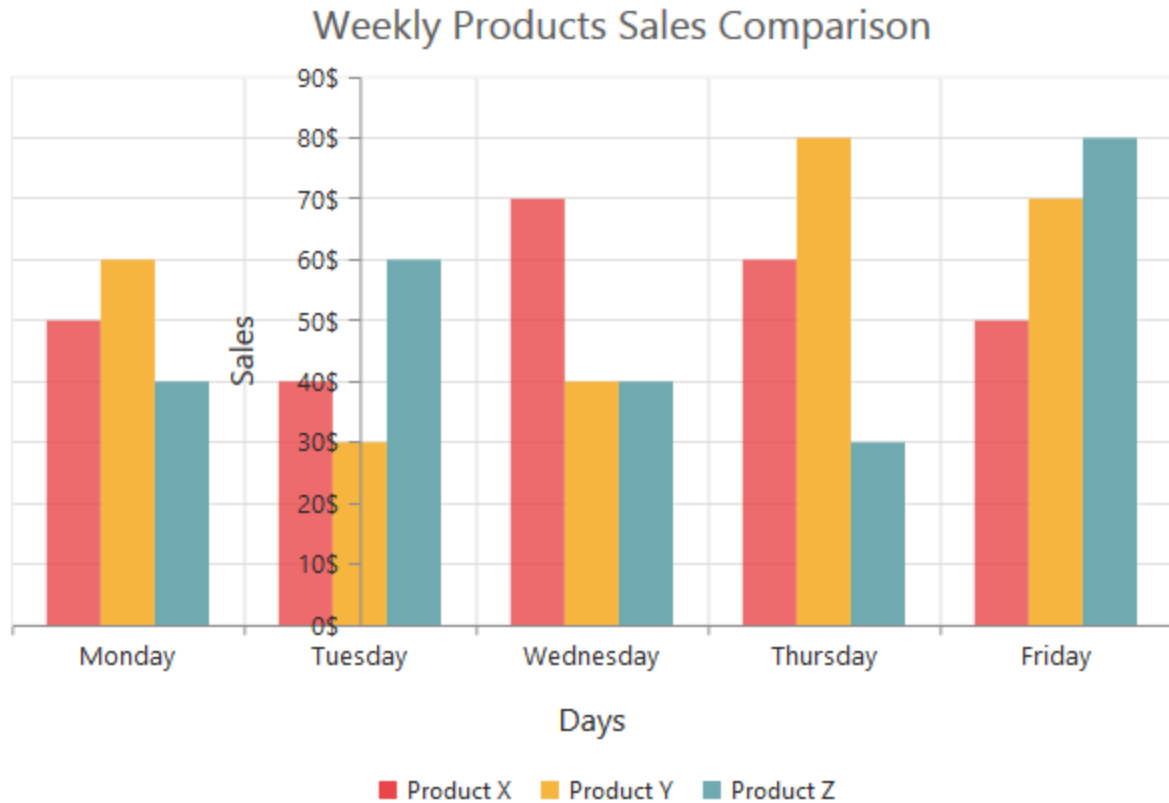
For crossing in a category type horizontal axis, either a string object or a number corresponding to the index of category value can be used for [crossesAt](#) property of vertical axes.

**Warning:** String value provided for [crossesAt](#) property is case-sensitive.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryYAxis:
  {
    //Crosses horizontal axis at category value 'Third'
    crossesAt: 'Tuesday',
    //...
  }
  //...
});
```





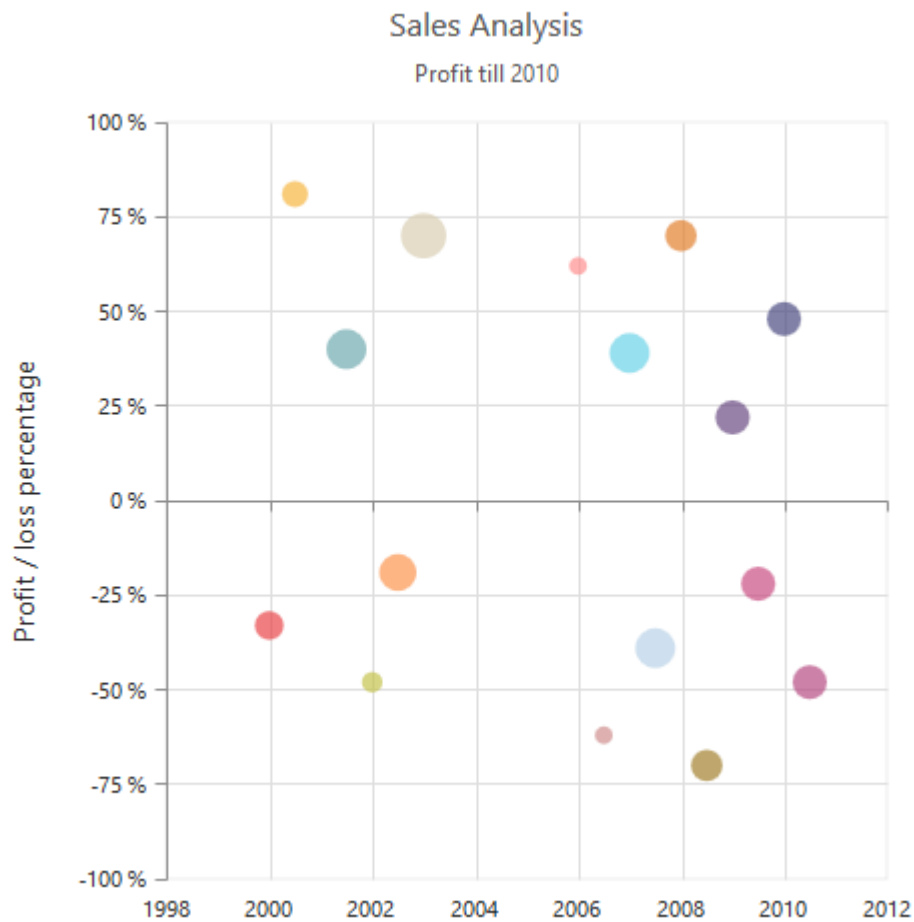
#### Positioning the axis elements while crossing

The [showNextToAxisLine](#) property is used for controlling the axis elements movement along with the axis line while axis crossing is performed. When the showNextToAxisLine is set as false only the axis line and the tick lines are placed at the crossing Value and the axis elements will be placed outside the chart area. The default value of [showNextToAxisLine](#) is **true**.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis:
  {
    //Crosses primary Y axis at 0
    crossesAt: 0,
    showNextToAxisLine: false,
    //...
  }
  //...
});
```

The axis is placed at the crossing value without the axis elements

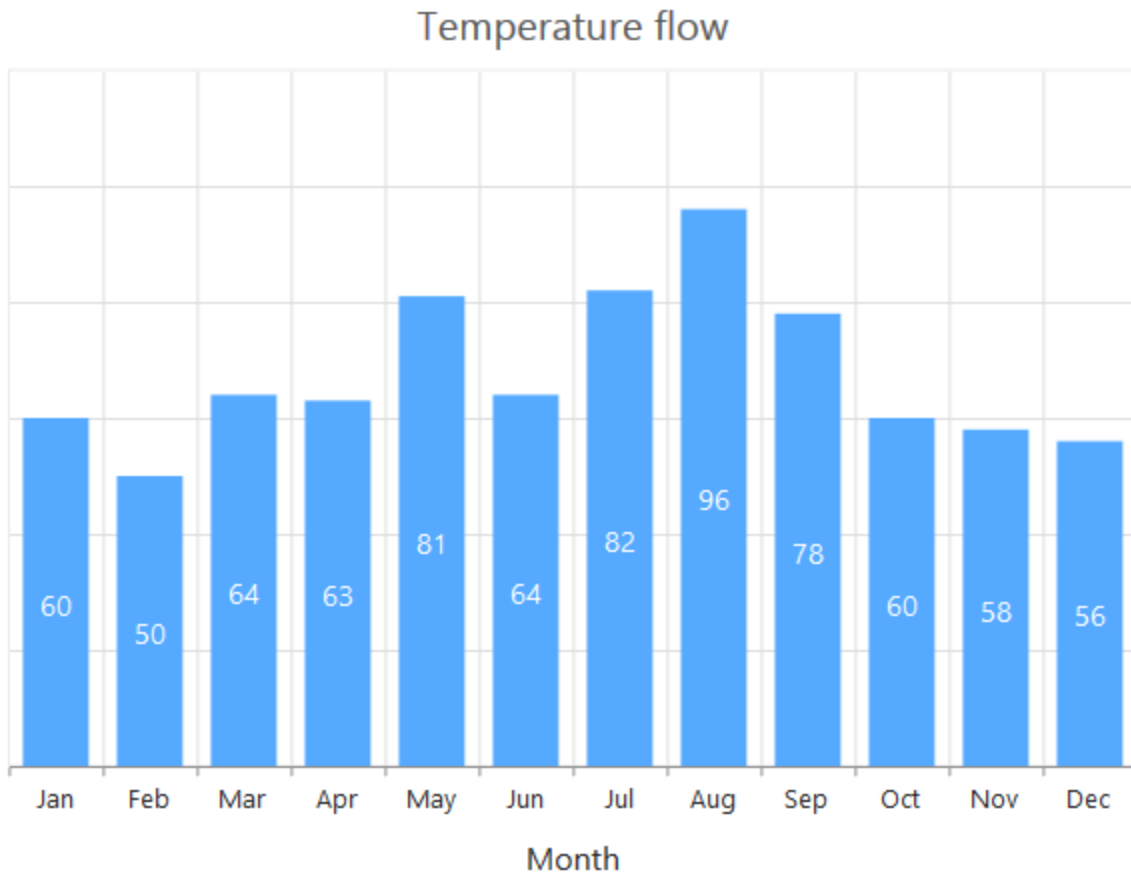


#### Axis Visibility

Axis visibility can be controlled by using the [visible](#) property of the axis. The default value of the [visible](#) property is **true**.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryYAxis: {
    //Disabling visibility of Y-axis
    visible: false
    // ...
  },
  // ...
});
```

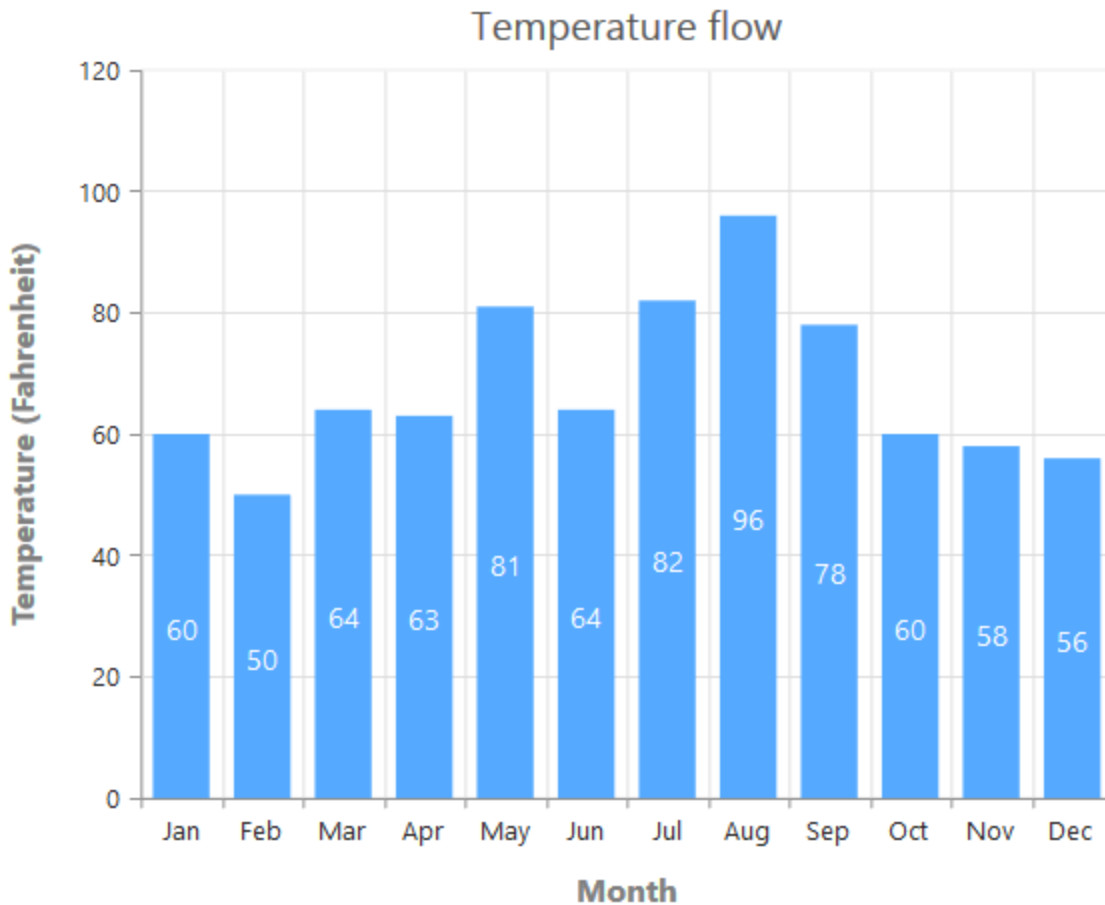


#### Axis title

The [title](#) property in the axis provides options to customize the text and font of the axis title. Axis does not display the title, by default. Title text can also be trimmed based on the title text length or specified length.

#### JAVASCRIPT

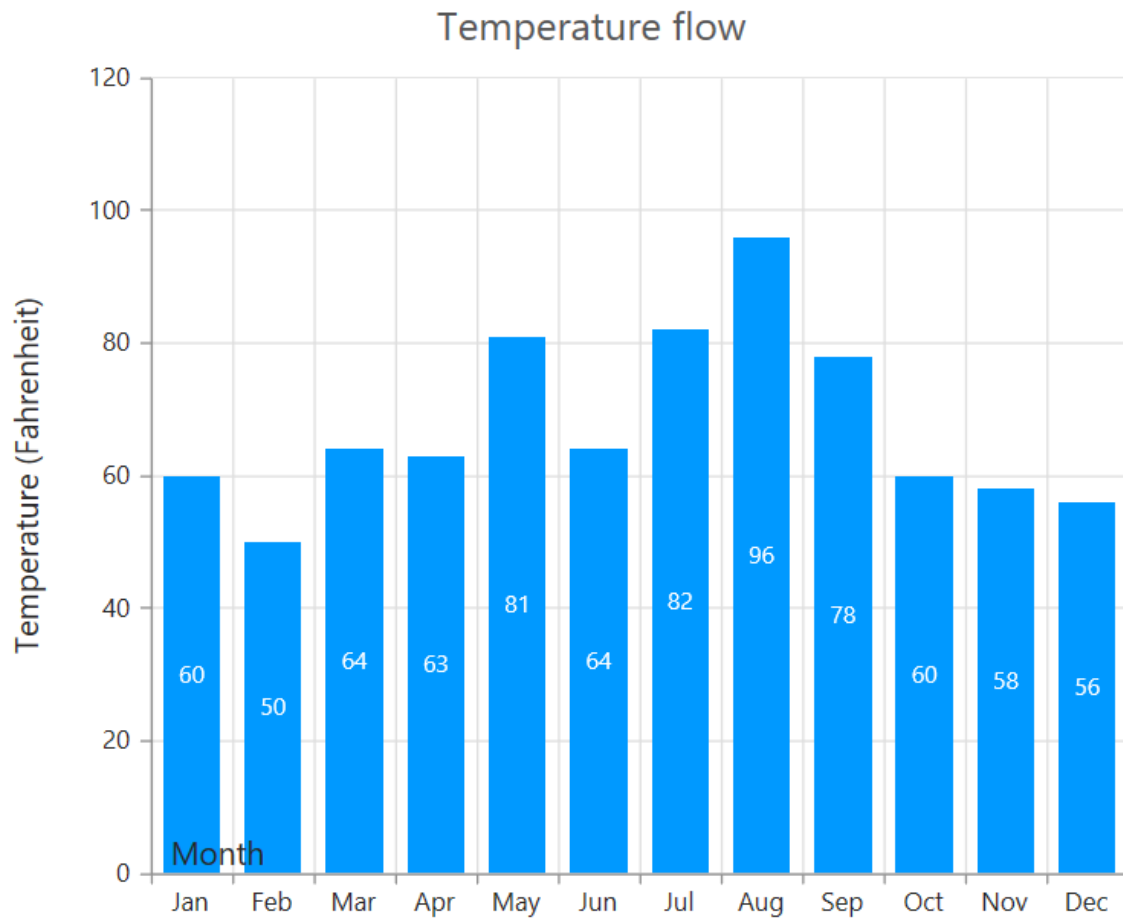
```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Customizing axis title
    title : {
      text : 'Month',
      font : {
        fontFamily : 'Segoe UI',
        size : '16px',
        fontWeight : 'bold' ,
        color : 'Grey',
      },
    },
    enableTrim : true,
    maximumTitleWidth : 80
  }
  // ...
},
  // ...
});
```



You can modify the position of the axis title either inside or outside the chart area using the property `[position]`. By default, it will be placed outside the chart area. In addition, you can also change the alignment of the title to near, far and center by `[alignment]` property, using `[offset]` property you can change the position with respect to pixels.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Customizing axis title
    title: {
      text: 'Month',
      position: 'inside',
      alignment: 'near',
      offset: 10
    }
    // ...
  },
  // ...
});
```

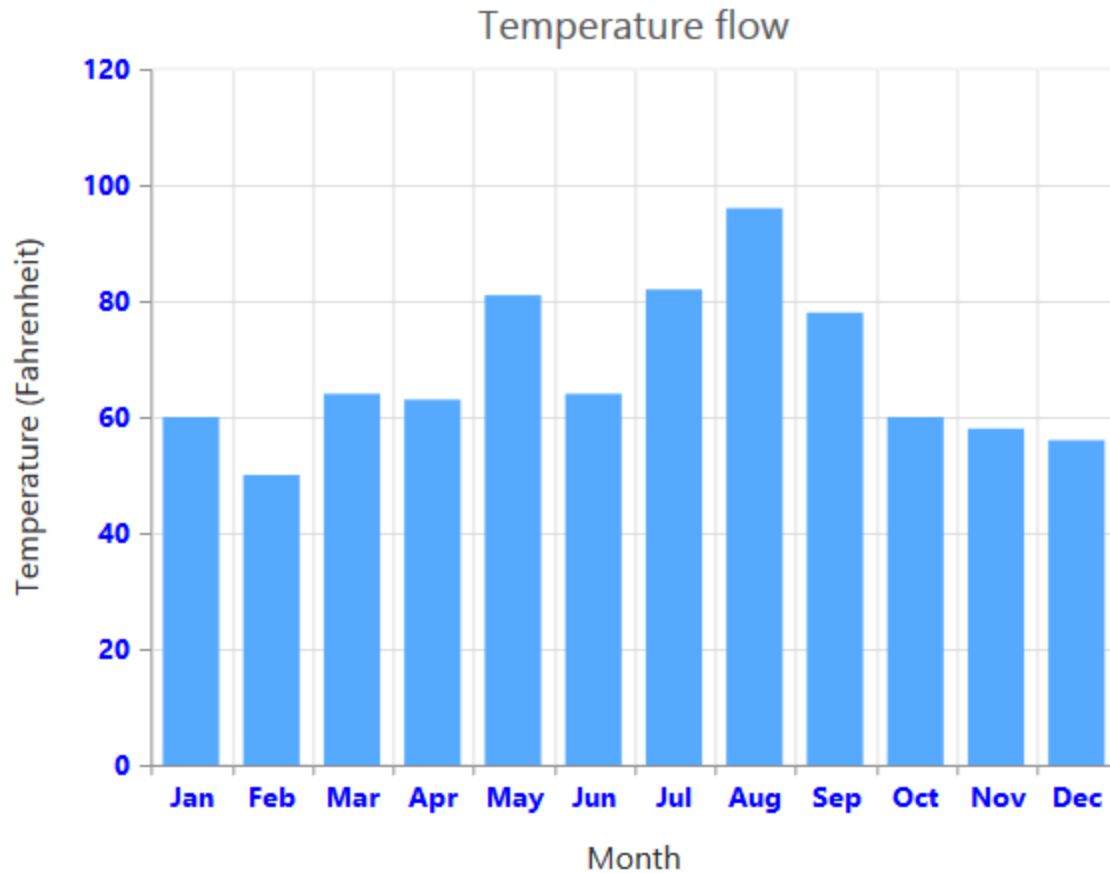


#### Label customization

The [font](#) property of the axis provides options to customize the [font-family](#), [color](#), [opacity](#), [size](#) and [font-weight](#) of the axis labels.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Customizing label appearance
    font : {
      fontFamily : 'Segoe UI',
      size : '14px',
      fontWeight : 'bold' ,
      color : 'blue',
    },
    // ...
  },
  // ...
});
```

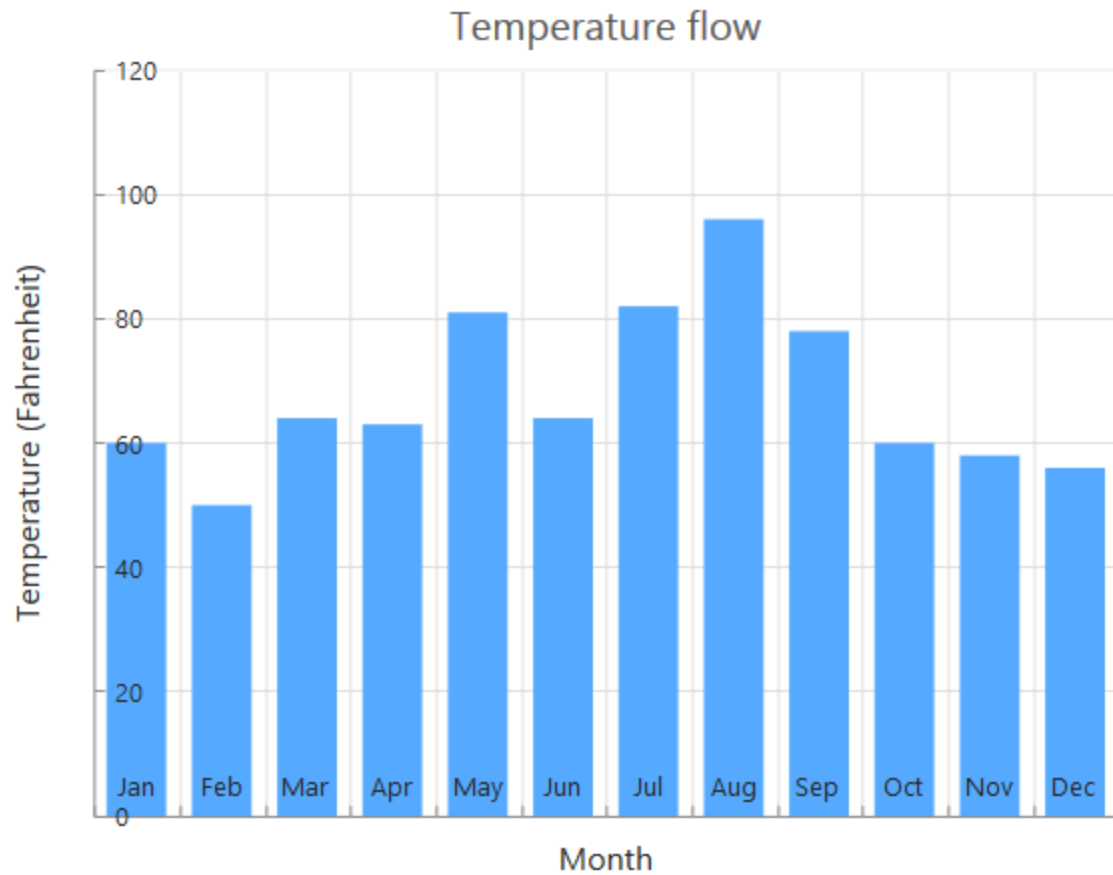


#### Label and tick positioning

Axis labels and ticks can be positioned inside or outside the chart area by using the [labelPosition](#) and [tickPosition](#) properties. The labels and ticks are positioned outside the chart area, by default.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Customizing label and tick positions
    labelPosition : 'inside',
    tickLinesPosition : 'inside',
    // ...
  },
  // ...
});
```



#### Edge labels placement

Labels with long text at the edges of an axis may appear partially outside the chart. The [edgeLabelPlacement](#) property can be used to avoid the partial appearance of the labels at the corners.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Customizing edge label placement
    edgeLabelPlacement : 'shift',
    // ...
  },
  // ...
});
```

Chart before setting edge label placement to X-axis

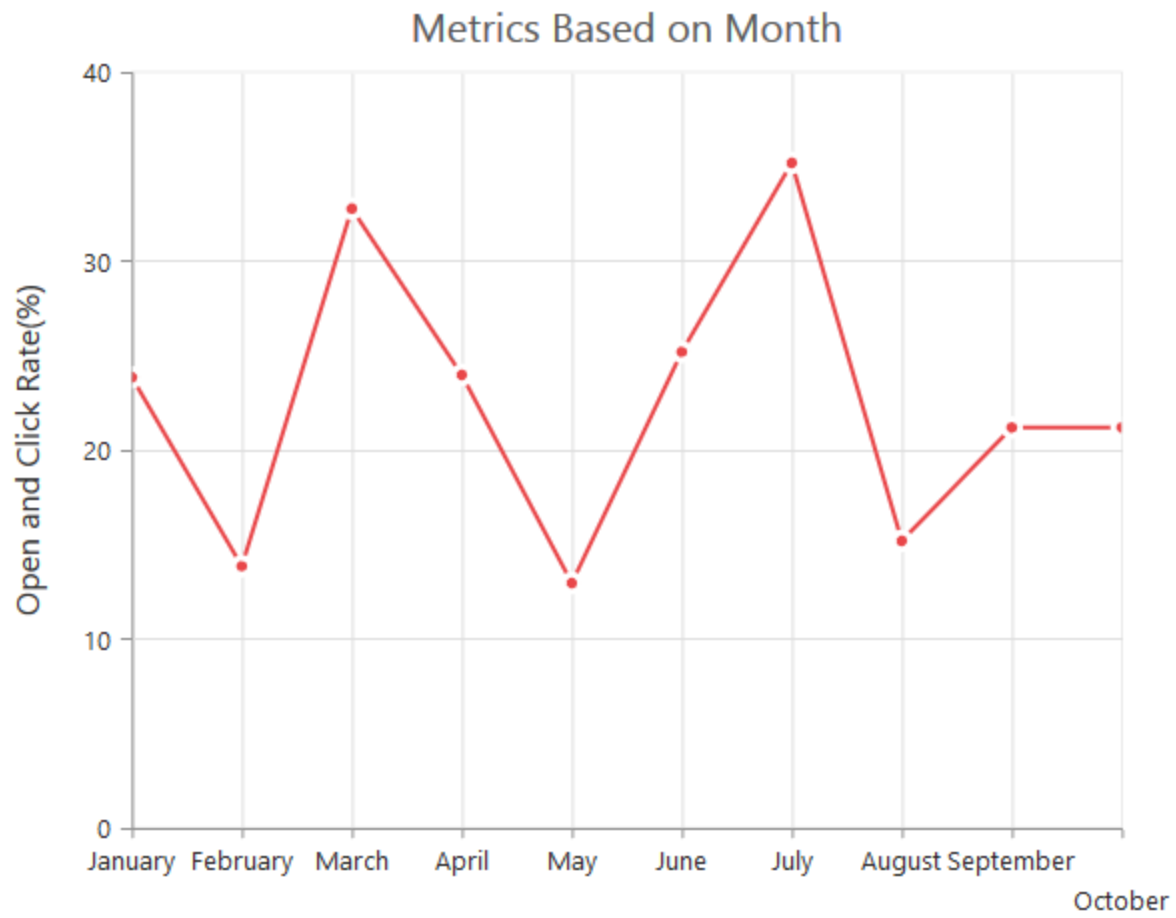
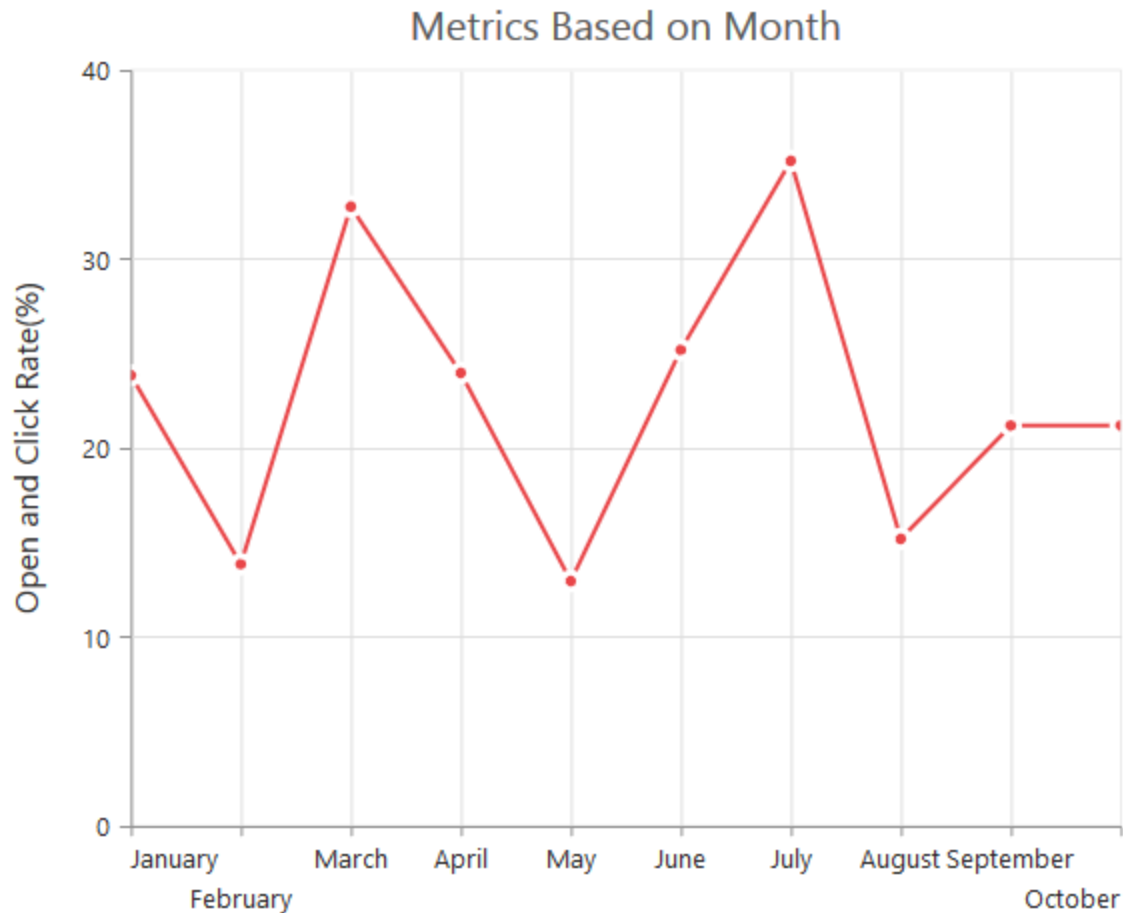


Chart after setting edge label placement to X-axis



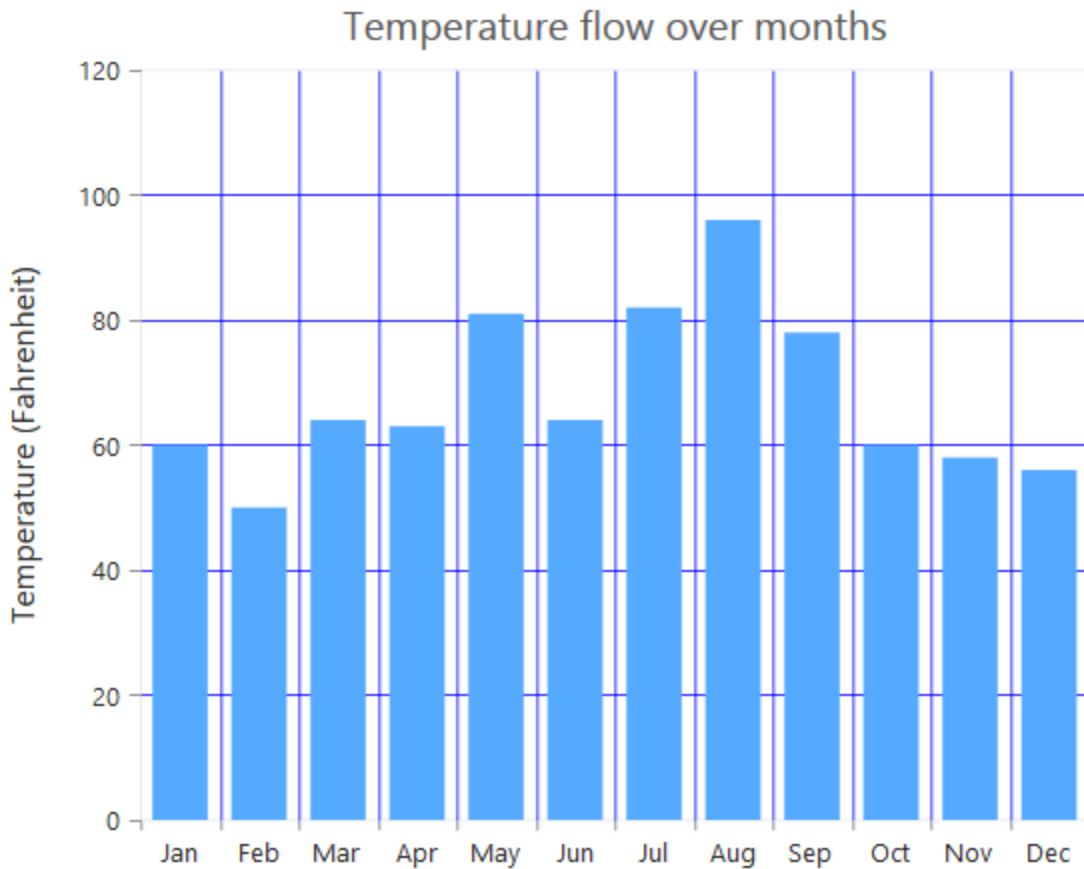


#### Grid lines customization

The [majorGridLines](#) and [minorGridLines](#) properties in the axis are used to customize the major grid lines and minor grid lines of an axis. They provide options to change the width, color, visibility and opacity of the grid lines. The minor grid lines are not visible, by default.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Customizing grid lines
    majorGridLines: {
      color: 'blue',
      visible: true,
      width: 1
    },
    minorTicksPerInterval: 0,
    minorGridLines: {
      color: 'red',
      visible: false,
      width: 1
    }
  }
  // ...
});
```

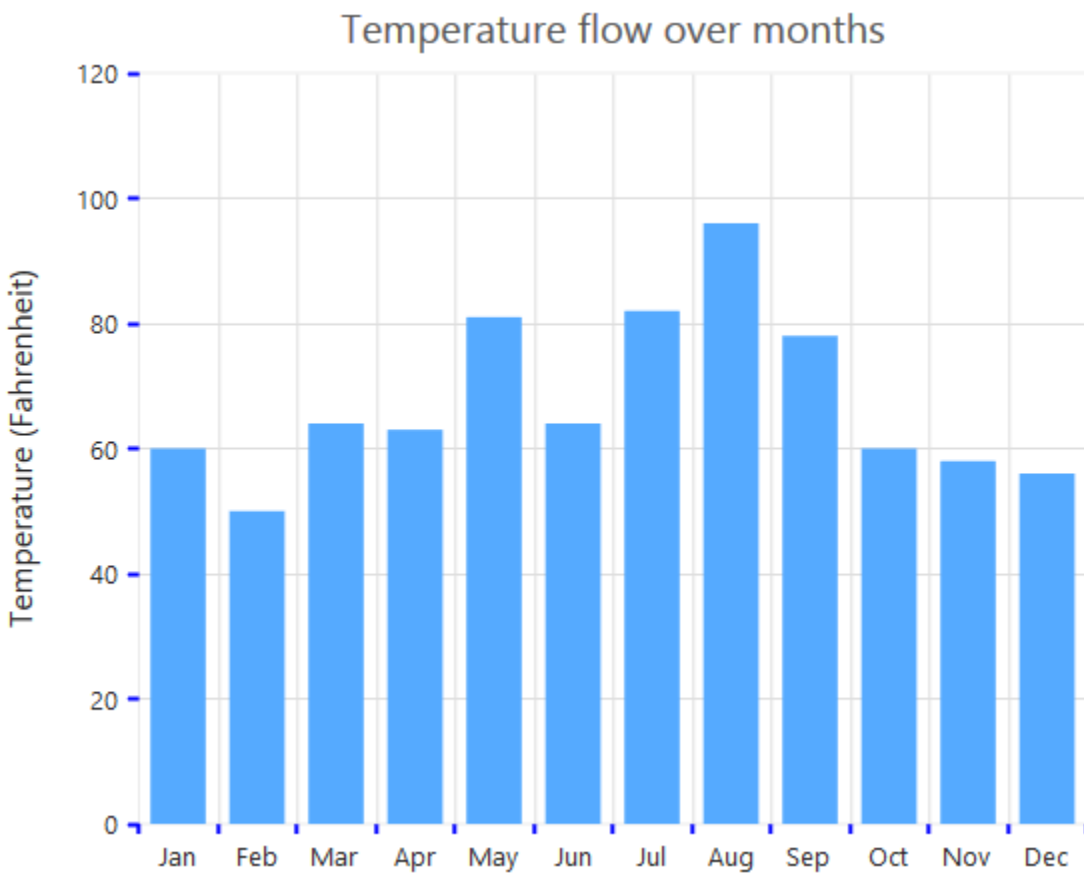


#### Tick lines customization

The [majorTickLines](#) and [minorTickLines](#) properties in the axis are used to customize the major tick lines of an axis and minor tick lines of an axis. They provide options to change the width, size, color and visibility of the grid lines. The minor tick lines are not visible, by default.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Customizing tick lines
    majorTickLines: {
      color: 'blue',
      visible: true,
      width: 1,
      size: 5,
    },
    minorTicksPerInterval: 0,
    minorTickLines: {
      color: 'red',
      visible: false,
      width: 1,
      size: 5,
    }
    // ...
  }
  // ...
});
```



#### Inverting axis

Axis can be inverted by using the [isInversed](#) property of the axis. The default value of the [isInversed](#) property is **false**.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Inverting the X-axis
    isInversed: false
    // ...
  },
  // ...
});
```

Chart before inverting the axes

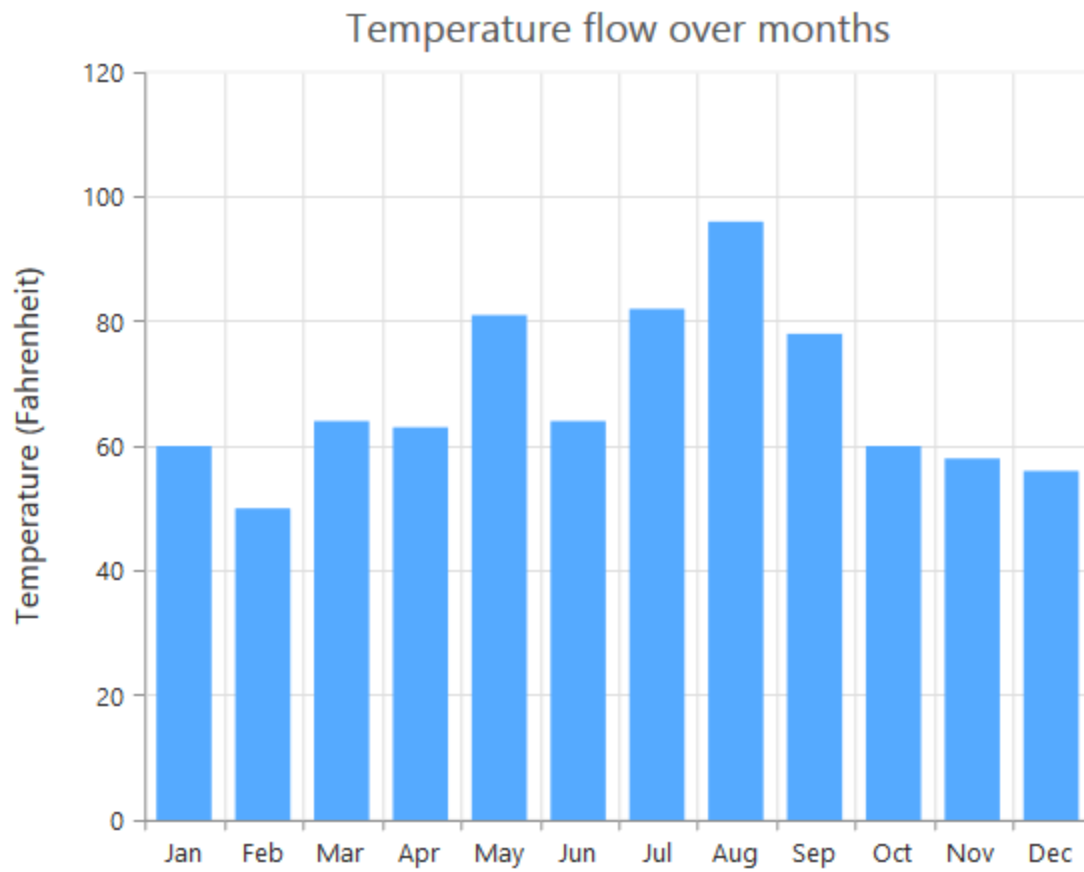
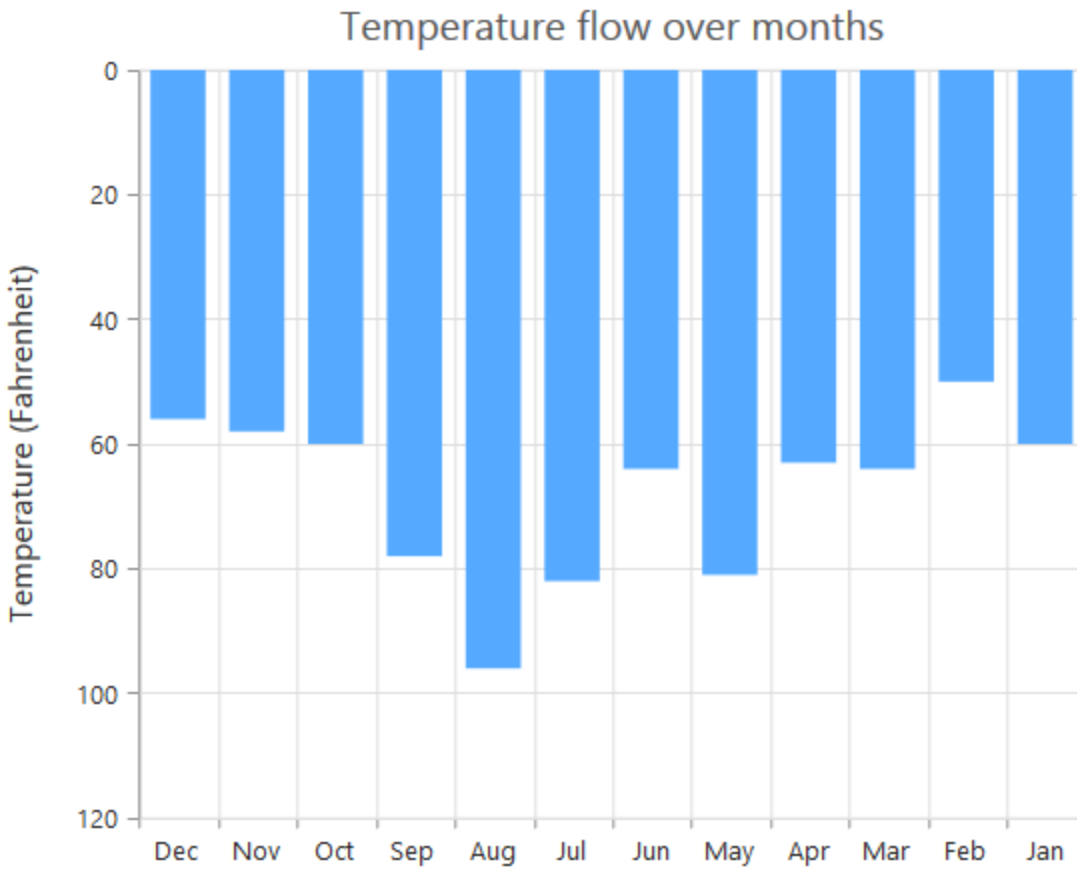


Chart after inversing the axes



*Place axes at the opposite side*

The [opposedPosition](#) property of axis can be used to place the axis at the opposite side of its default position. The default value of the [opposedPosition](#) property is **false**.

#### **JAVASCRIPT**

```
$("#chartContainer").ejChart({
  primaryXAxis: {
    //Placing axis at the opposite side of its normal position
    opposedPosition: true
    // ...
  },
  // ...
});
```

#### **Chart with X and Y axes at normal position**

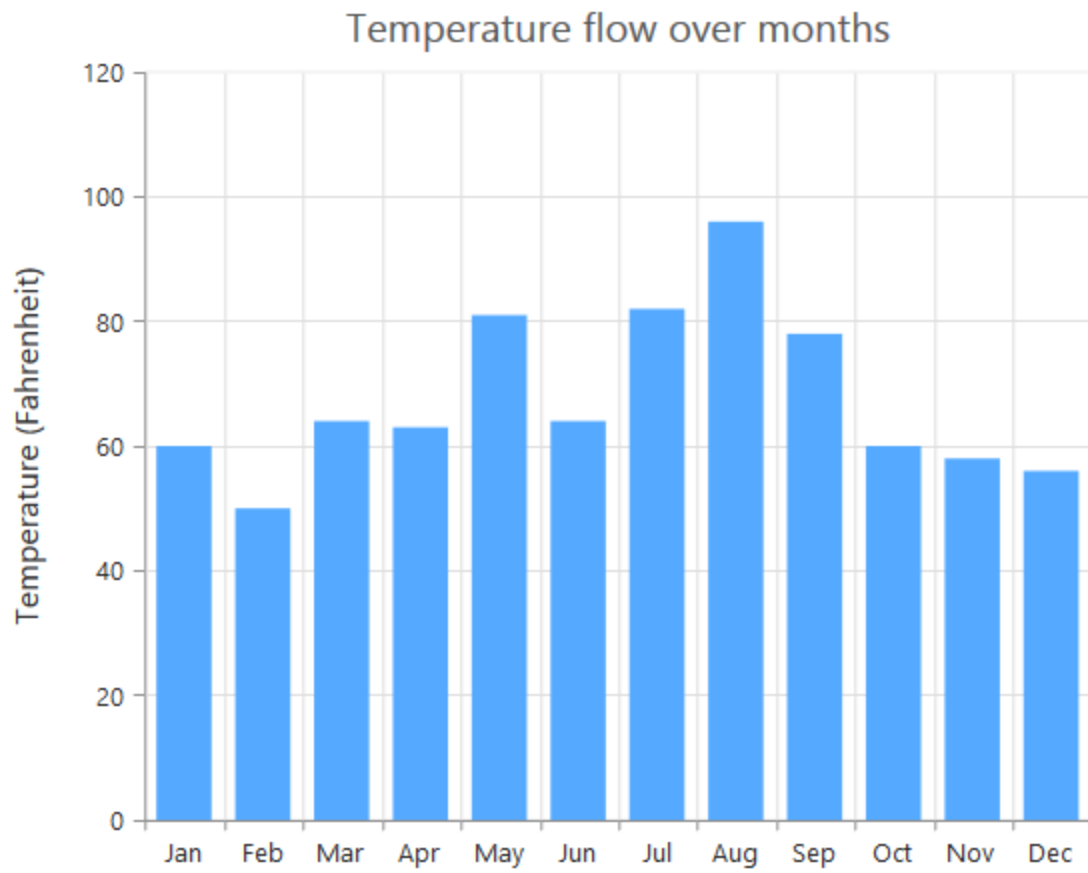
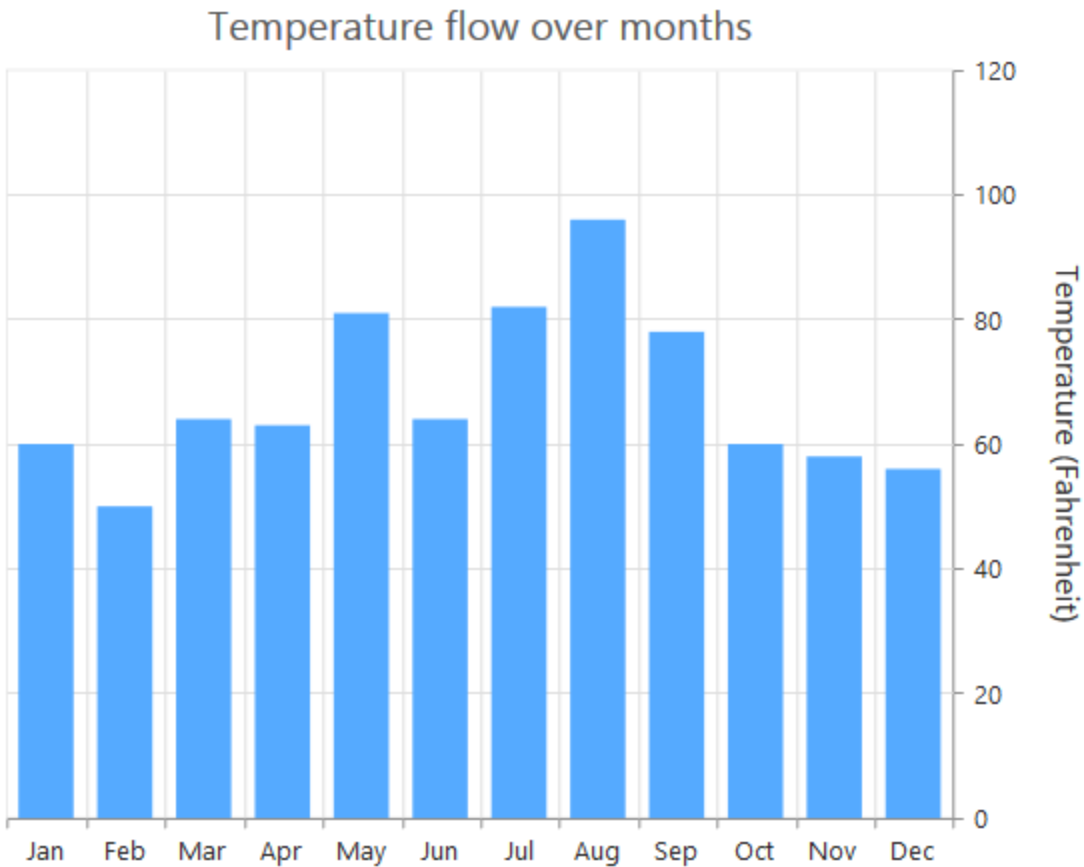


Chart with Y-axis at opposed position



#### Maximum number of labels per 100 pixels

A maximum of 3 labels are displayed for each 100 pixels in the axis, by default. The maximum number of labels that is present within the 100 pixels length can be customized by using the [maximumLabels](#) property of the axis. This property is applicable only for an automatic range calculation and it does not work when you set the value for [interval](#) property in the axis range.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  primaryXAxis: {
    //Maximum number of labels per 100 pixels
    maximumLabels : 1
    // ...
  },
  // ...
});
```

#### Chart before setting maximum labels per 100 pixels

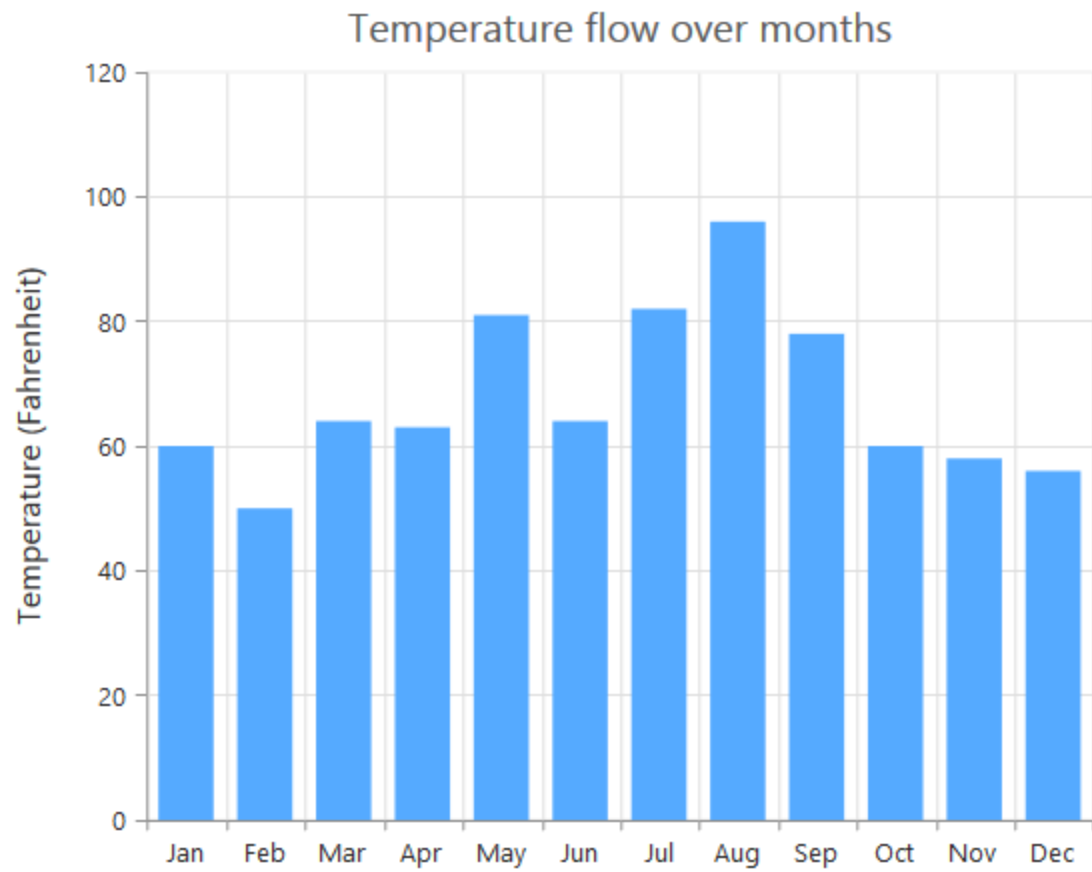
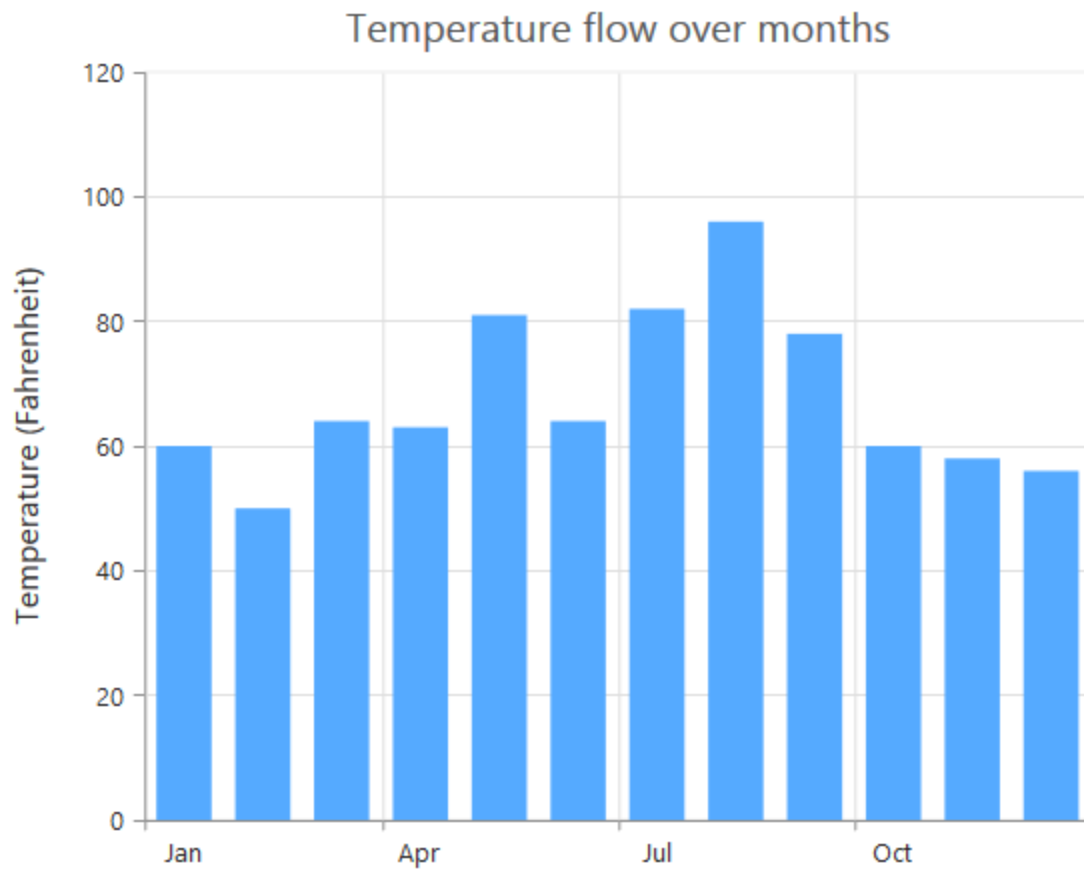


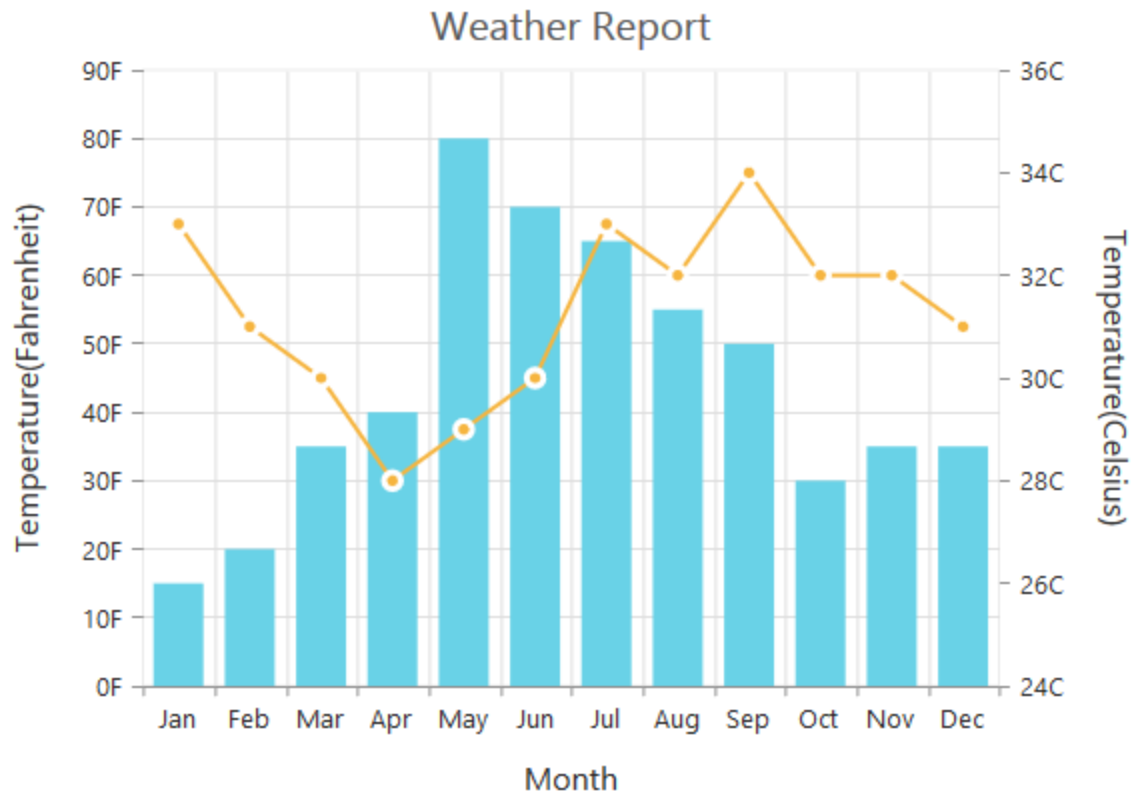
Chart after setting maximum labels one per 100 pixels





#### Multiple Axis

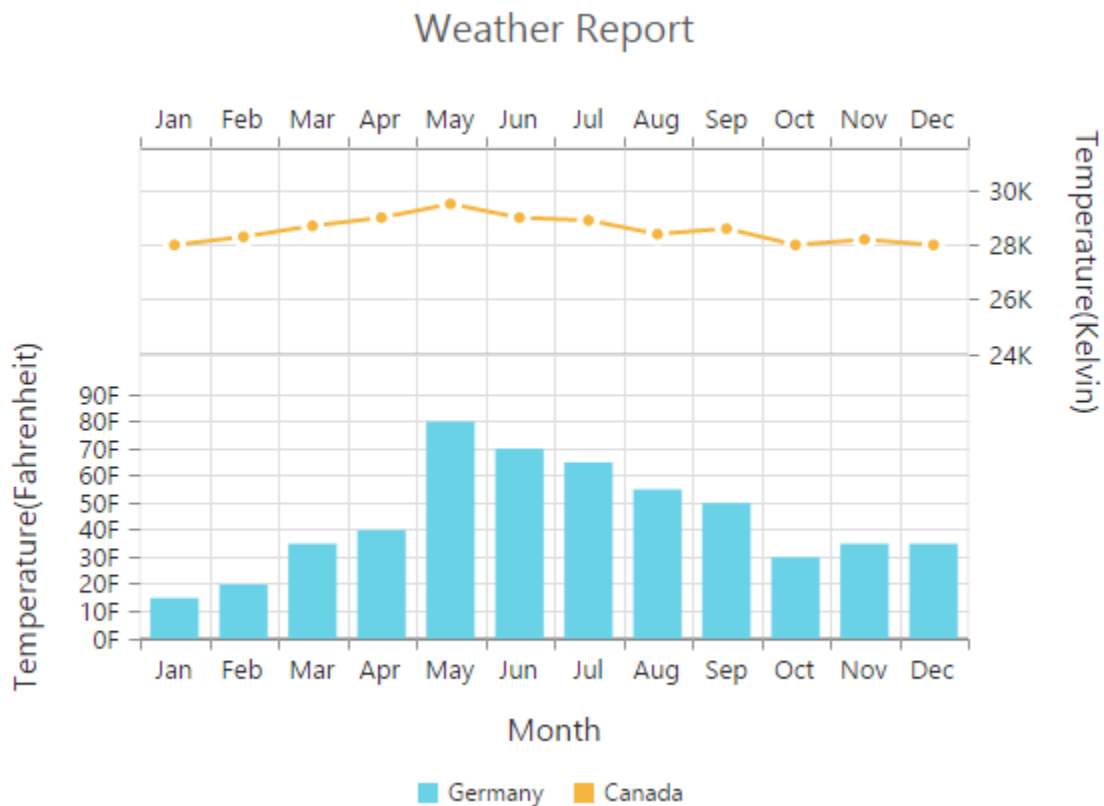
Multiple axes can be used in the Chart and chart area can be split into multiple panes to draw multiple series with multiple axes.



An additional horizontal or vertical axis can be added to the chart by adding an axis instance to the **axes** collection and then you can associate it to a series by specifying the name of the axis to the [xAxisName](#) or [yAxisName](#) property of the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  // Creating a secondary horizontal axis
  axes: [{
    name : 'SecondaryX',
    // ...
  }, {
    name : 'SecondaryY',
    // ...
  }],
  // ...
  series: [{
    // Binding secondary X-axis with a series
    xAxisName : 'SecondaryX',
    // Binding secondary Y-axis with a series
    yAxisName : 'SecondaryY',
    // ...
  },
  {
    // ...
  }],
  // ...
});
```

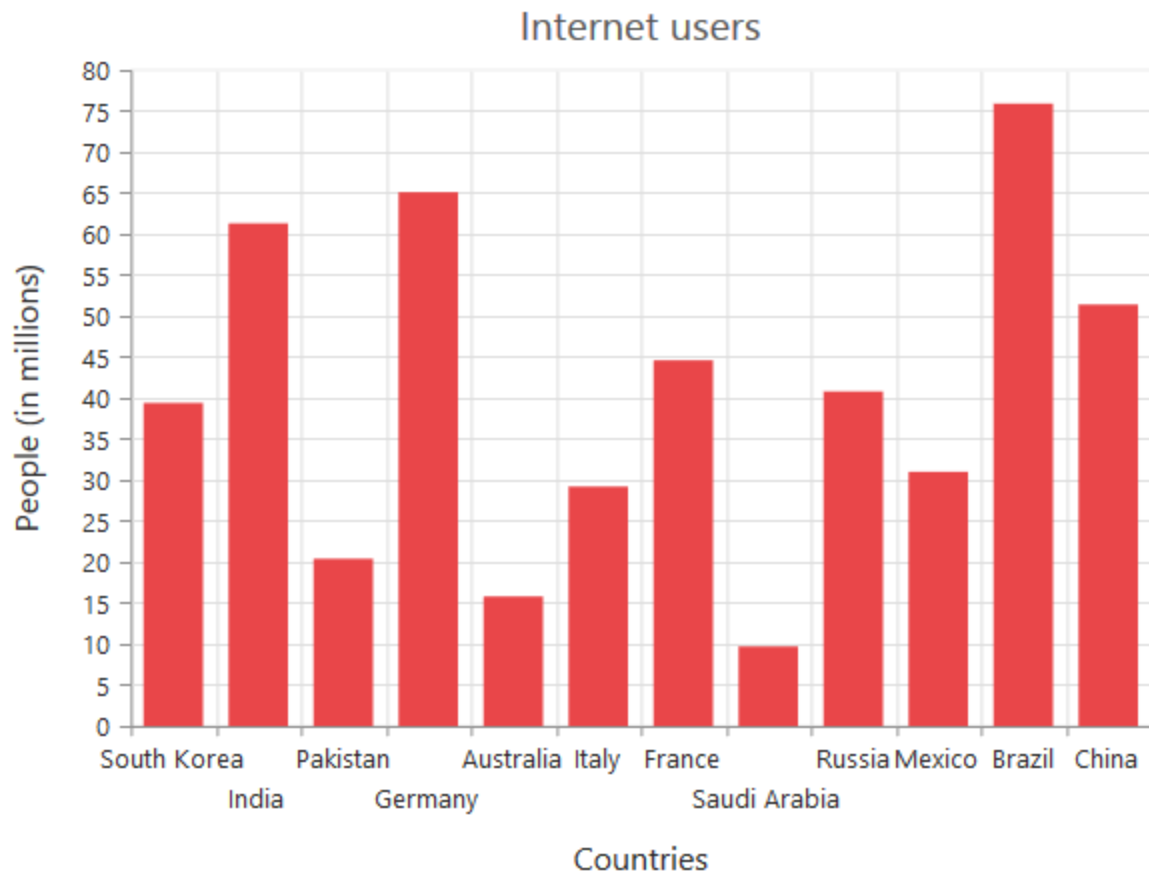


#### Smart Axis Labels

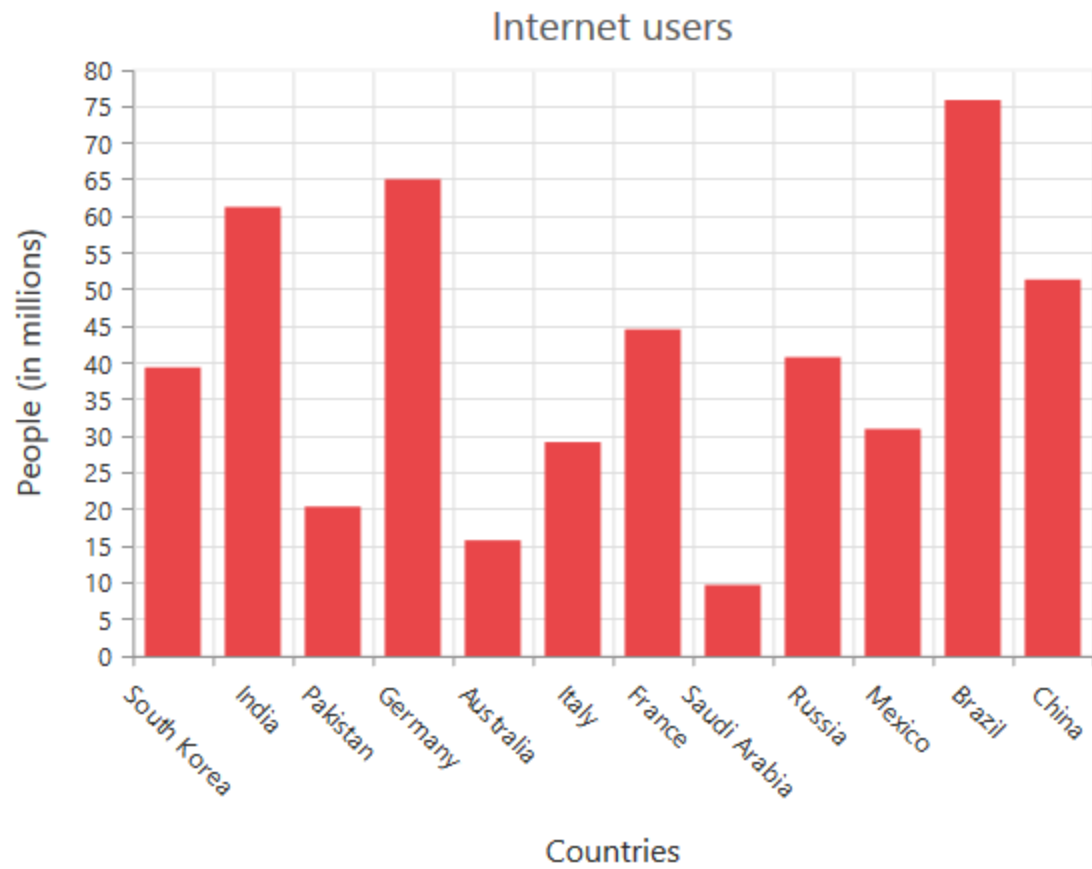
When the Axis labels overlap with each other based on the chart dimensions and label size, you can use the [labelIntersectAction](#) property of the axis to avoid overlapping. The default value of the [labelIntersectAction](#) is **none**. The other available values of the Label Intersect Actions are **rotate45**, **rotate90**, **trim**, **multipleRows**, **wrap**, **wrapByWord** and **hide**.

#### JAVASCRIPT

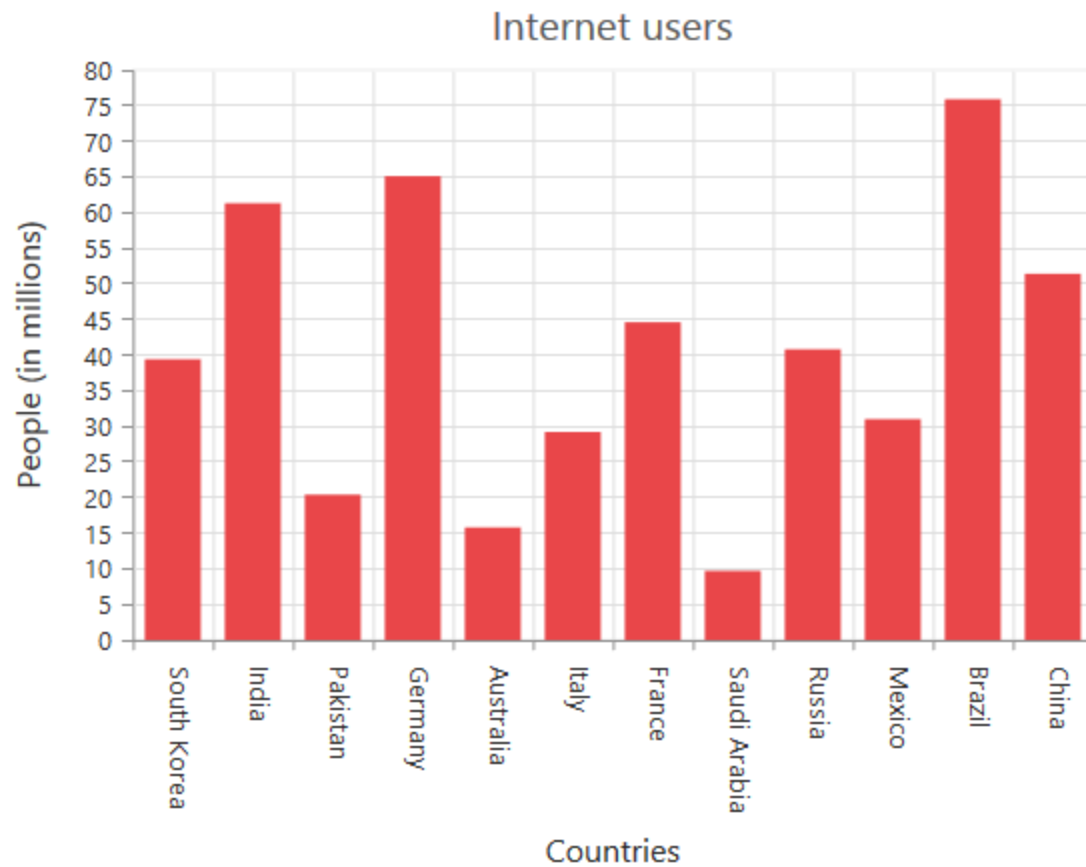
```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // Avoid overlapping of x-axis labels
  primaryXAxis: {
    labelIntersectAction : 'multipleRows',
    // ...
  },
  // ...
});
```



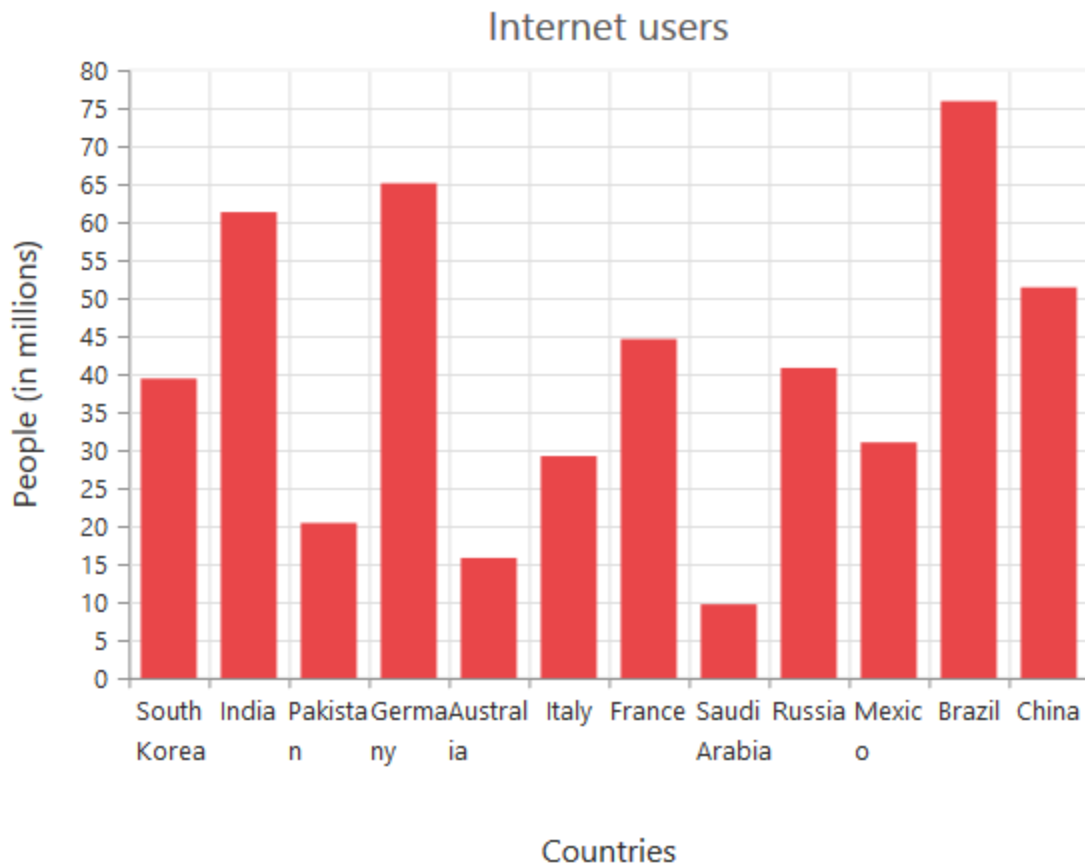
The following screenshot displays the result, when the [labelIntersectAction](#) property is set as **rotate45**.



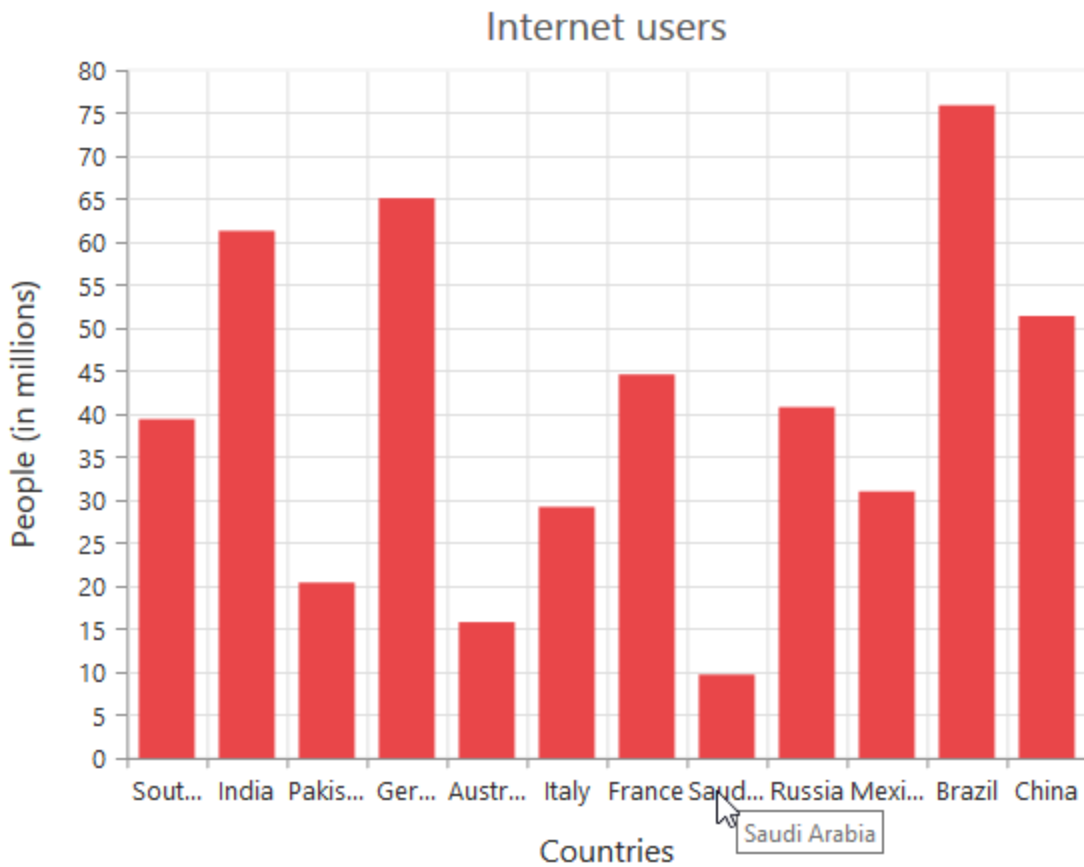
The following screenshot displays the result, when the [labelIntersectAction](#) property is set as **rotate90**.



The following screenshot displays the result, when the [labelIntersectAction](#) property is set as **wrap**.

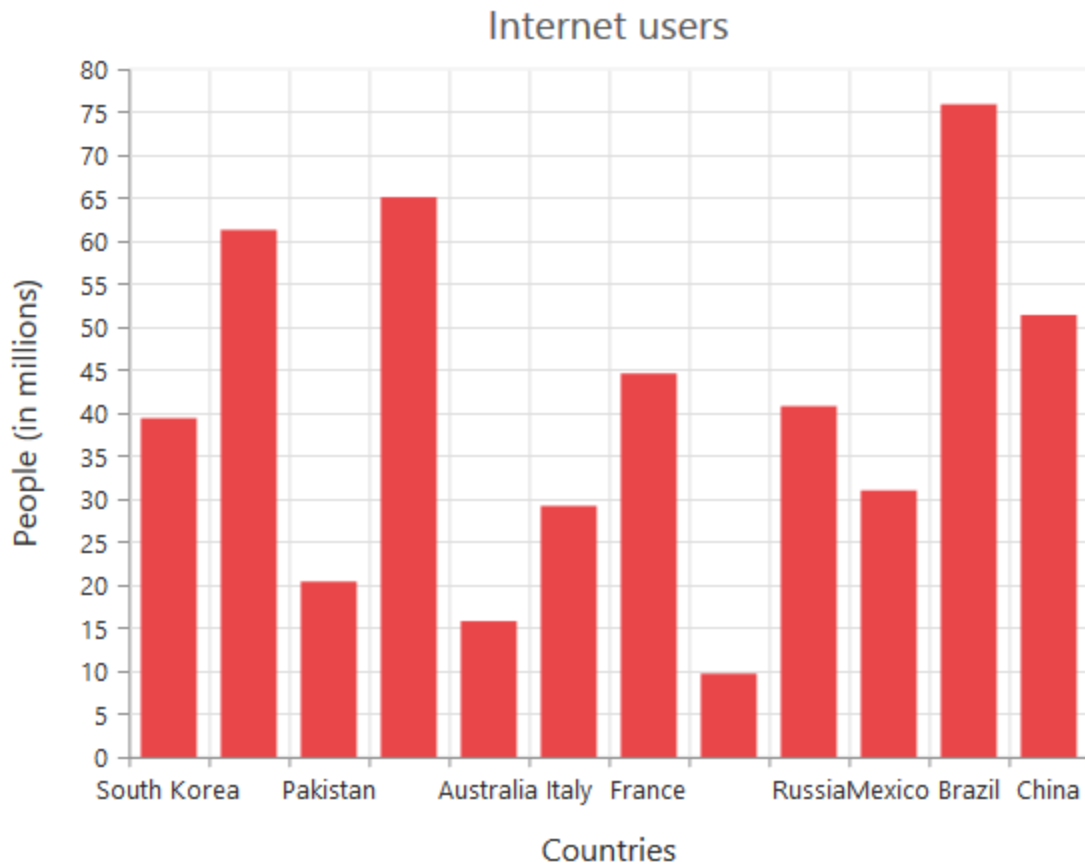


The following screenshot displays the result, when of setting the **trim** as value to the [labelIntersectAction](#) property.

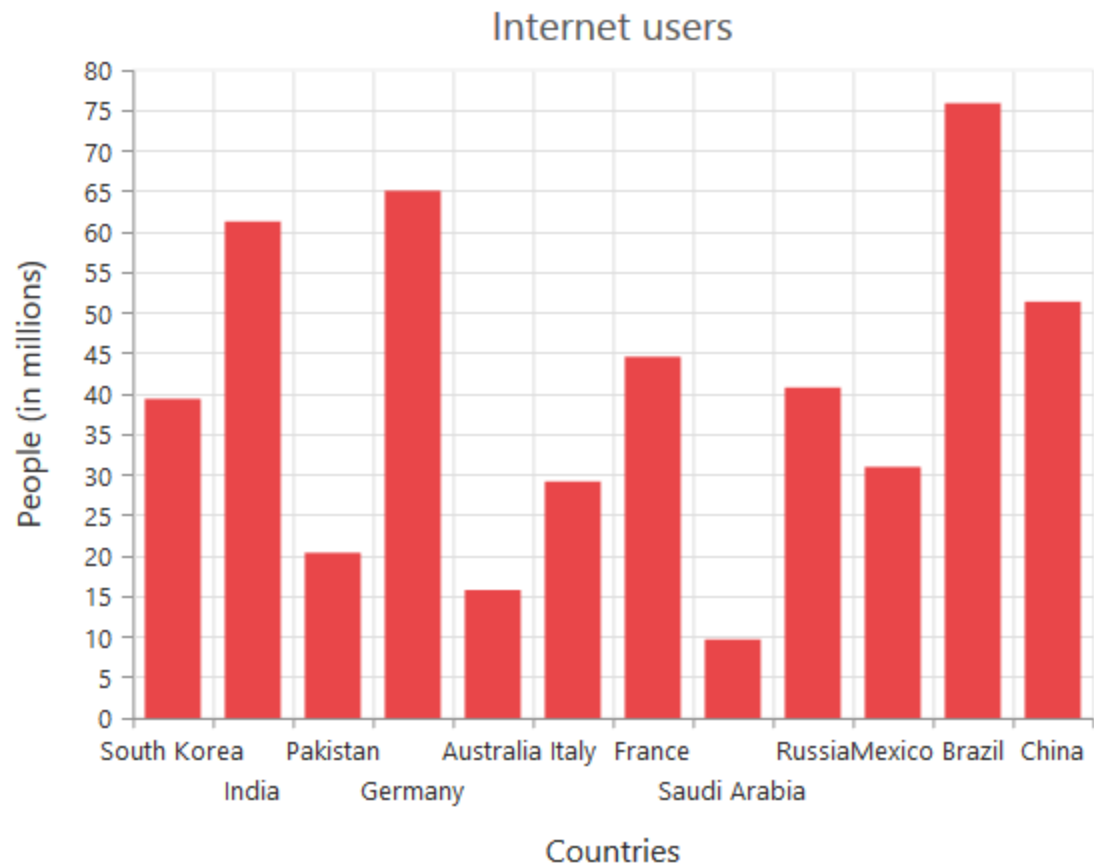


The following screenshot displays the result, when the [labelIntersectAction](#) property is set as **hide**.

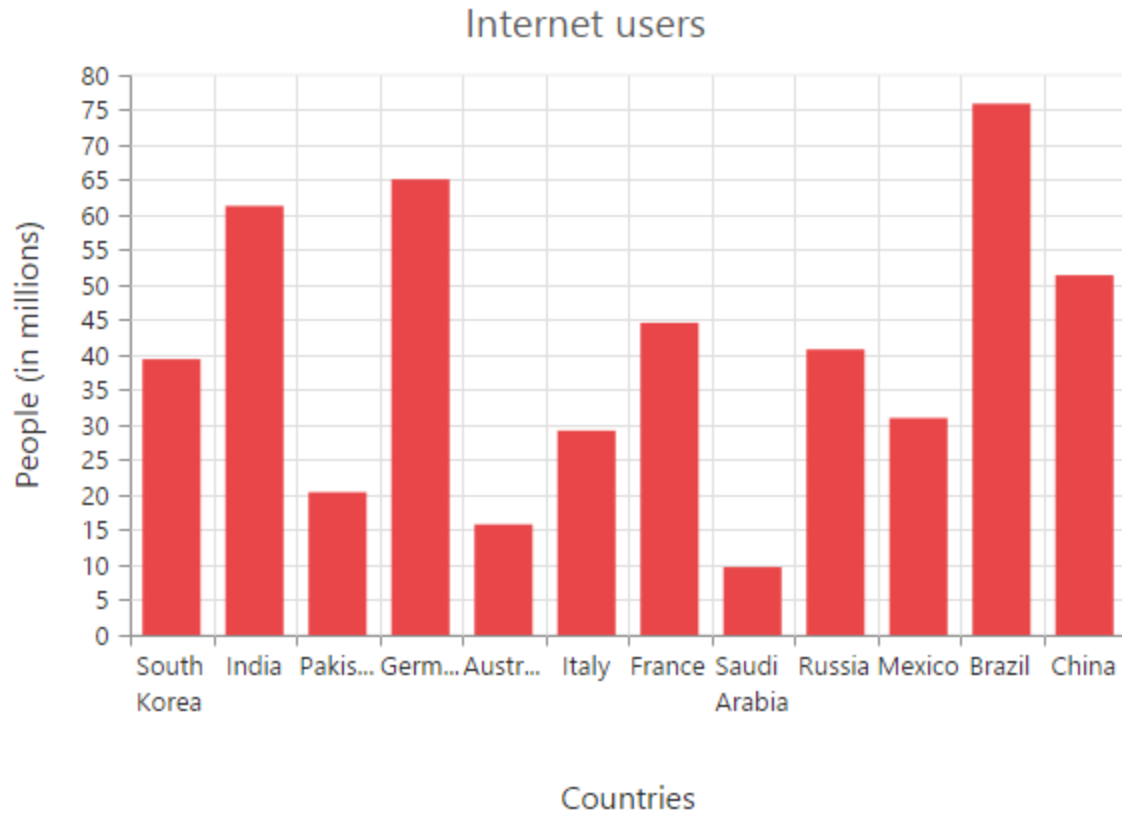




The following screenshot displays the result, when the [labelIntersectAction](#) property is set as **multipleRows**.



The following screenshot displays the result, when the [labelIntersectAction](#) property is set as **wrapByWord**.

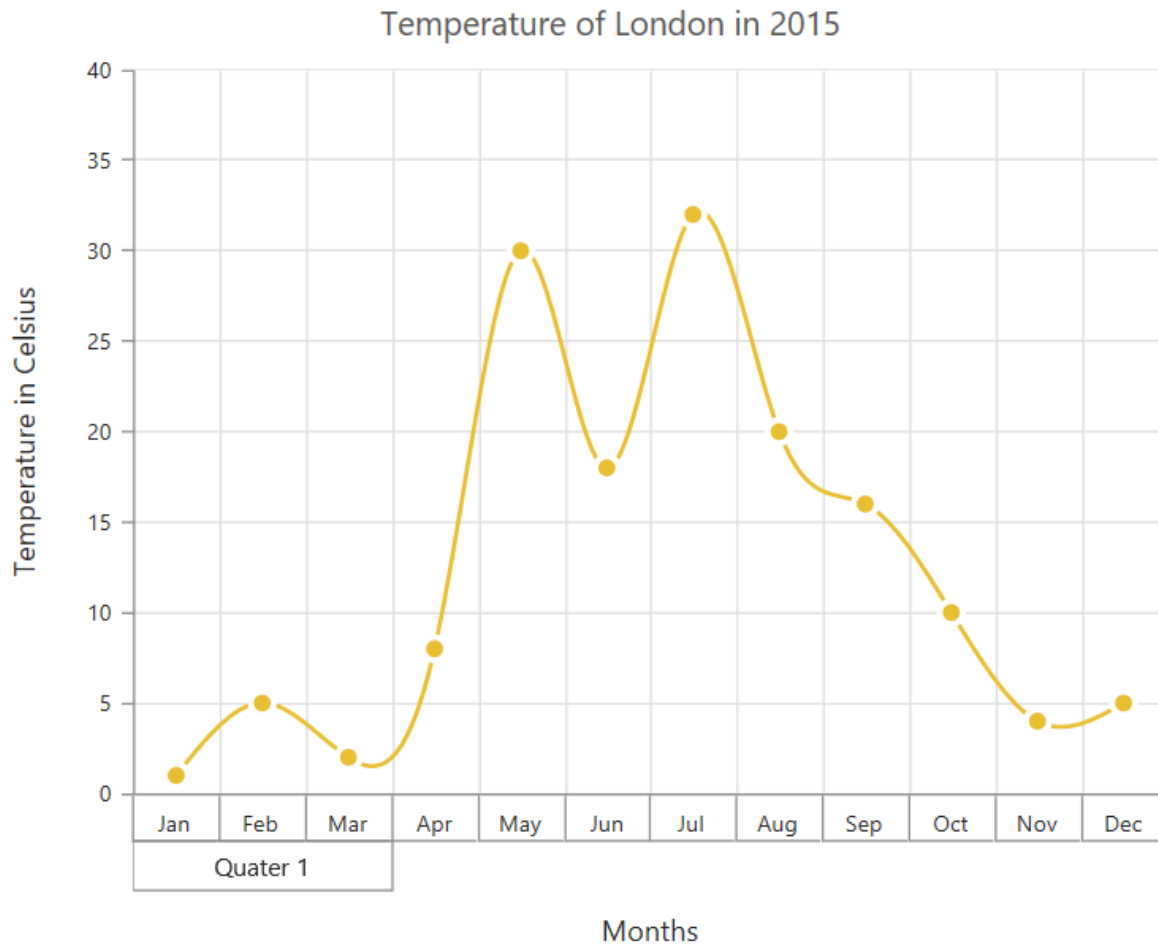


### Multi-level Labels

Axis can be customized with multiple levels of labels using the `[multiLevelLabels]` property. These labels are placed based on the start and end range values and we can add any number of labels to an axis.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  {
    primaryXAxis:
    {
      multiLevelLabels: [
        {
          visible: true,
          start: -0.5,
          end: 2.5,
          text: "Quater1"
        }
      ]
    }
  }
});
```

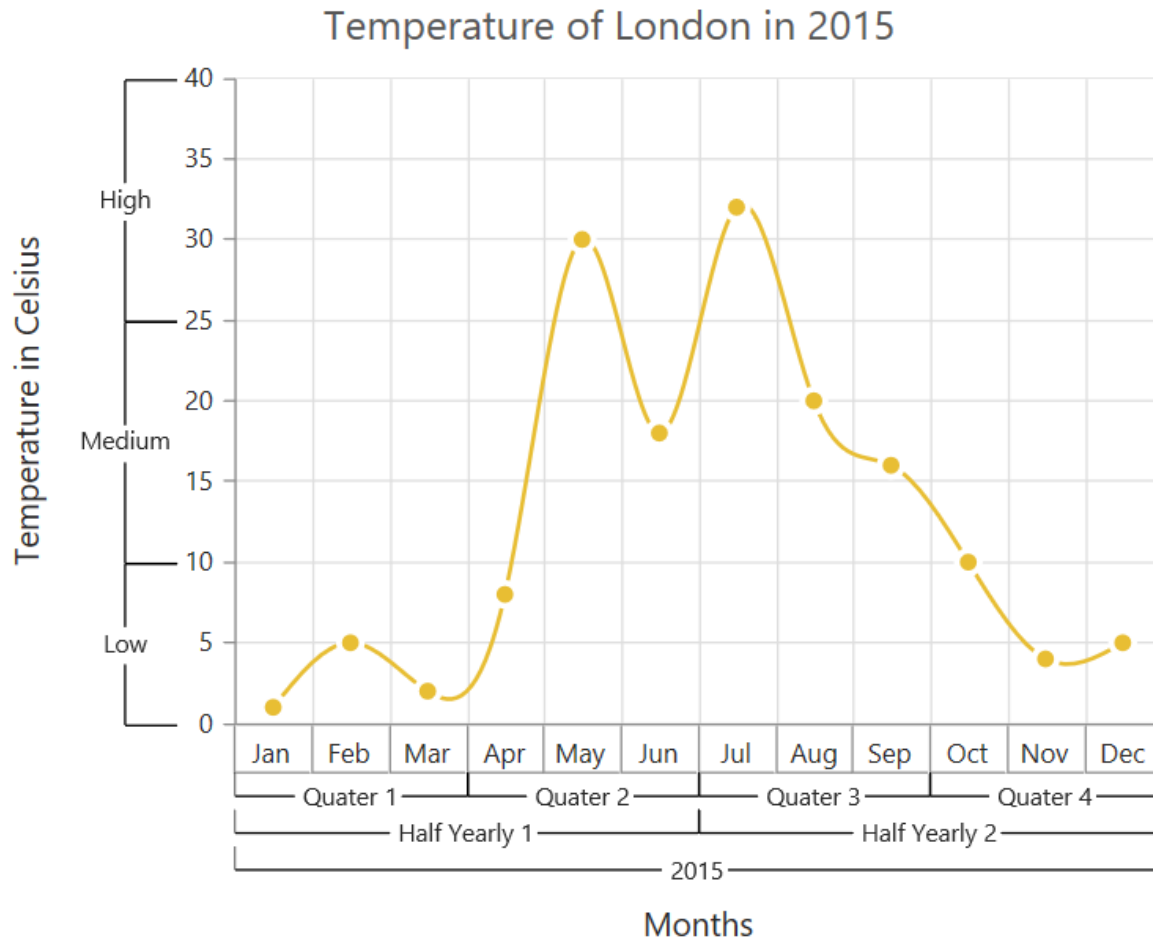


#### Customizing the multi-Level labels

The color, width and type of the border can be customized. The default border type is [Rectangle]. And the other supported border types are namely brace, curly brace, without top/bottom border and none.

#### JAVASCRIPT

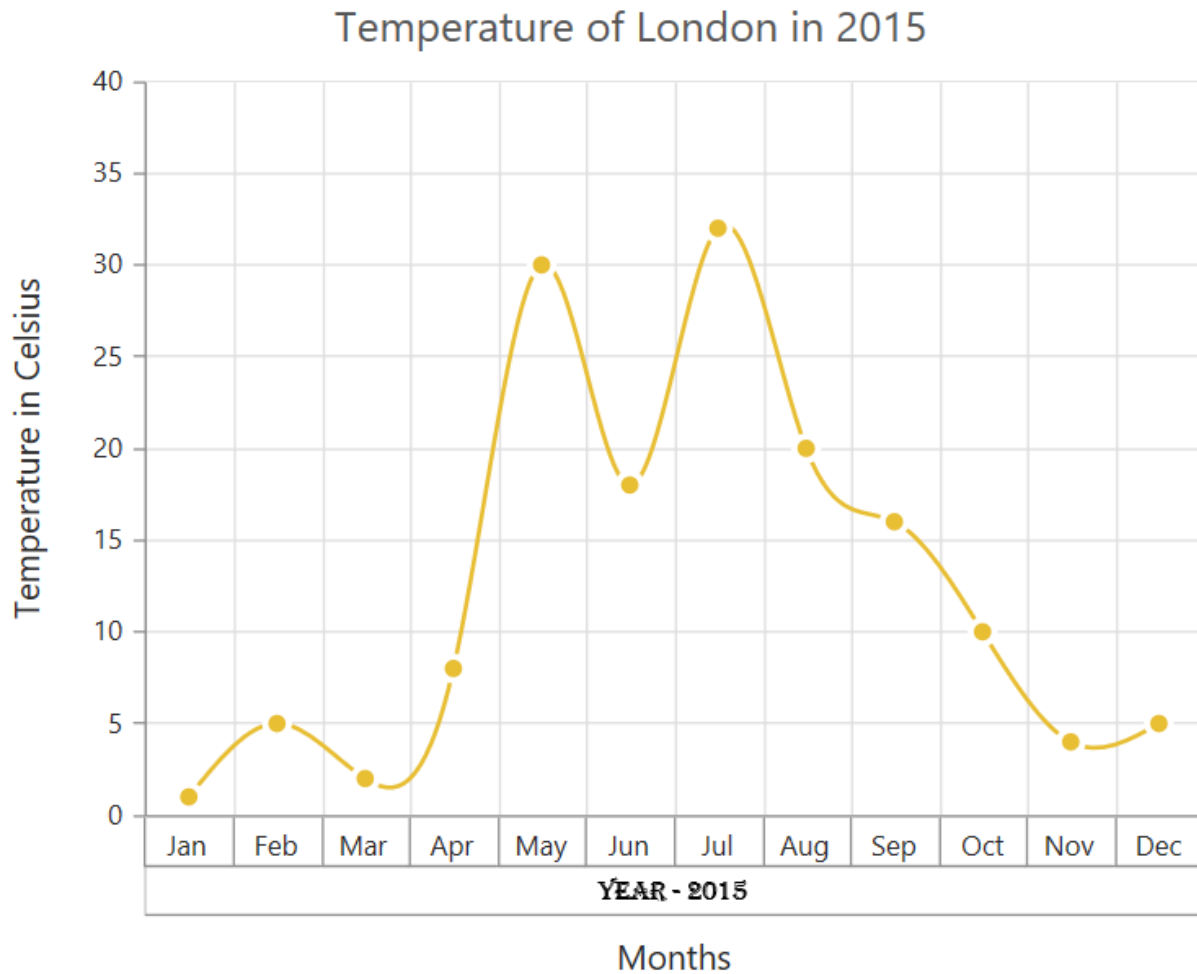
```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  {
    primaryXAxis:
    {
      multiLevelLabels: [
        {
          // customizing the border properties
          border:{
            type: "brace",
            width: 2,
            color: "black"
          }
        }
      ]
    }
  }
});
```



The text of the labels can be customized using the [text] and [font] properties

#### JAVASCRIPT

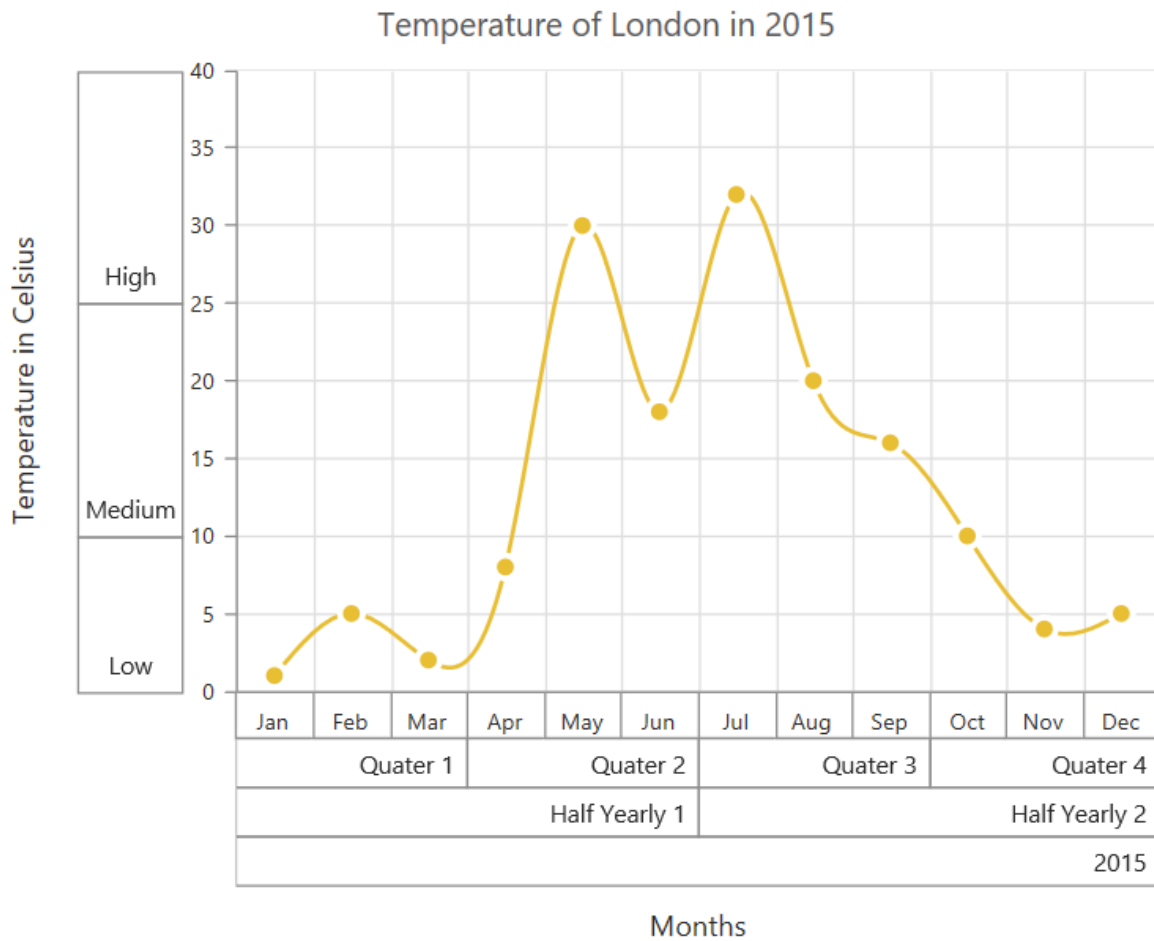
```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  {
    primaryXAxis:
    {
      multiLevelLabels: [
        {
          // customizing the text and font properties
          text: "Year - 2015",
          font:{
            fontFamily: "Algerian",
            size: "12px",
            color: "black"
          }
        }
      ]
    }
  }
});
```



You can change the alignment of the text to far, near and center position using the `textAlignment` property. By default, the text will be center aligned.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  {
    primaryXAxis:
    {
      multiLevelLabels: [
        {
          // customizing the text alignment
          textAlignment: "far",
        }
      ]
    }
  }
});
```

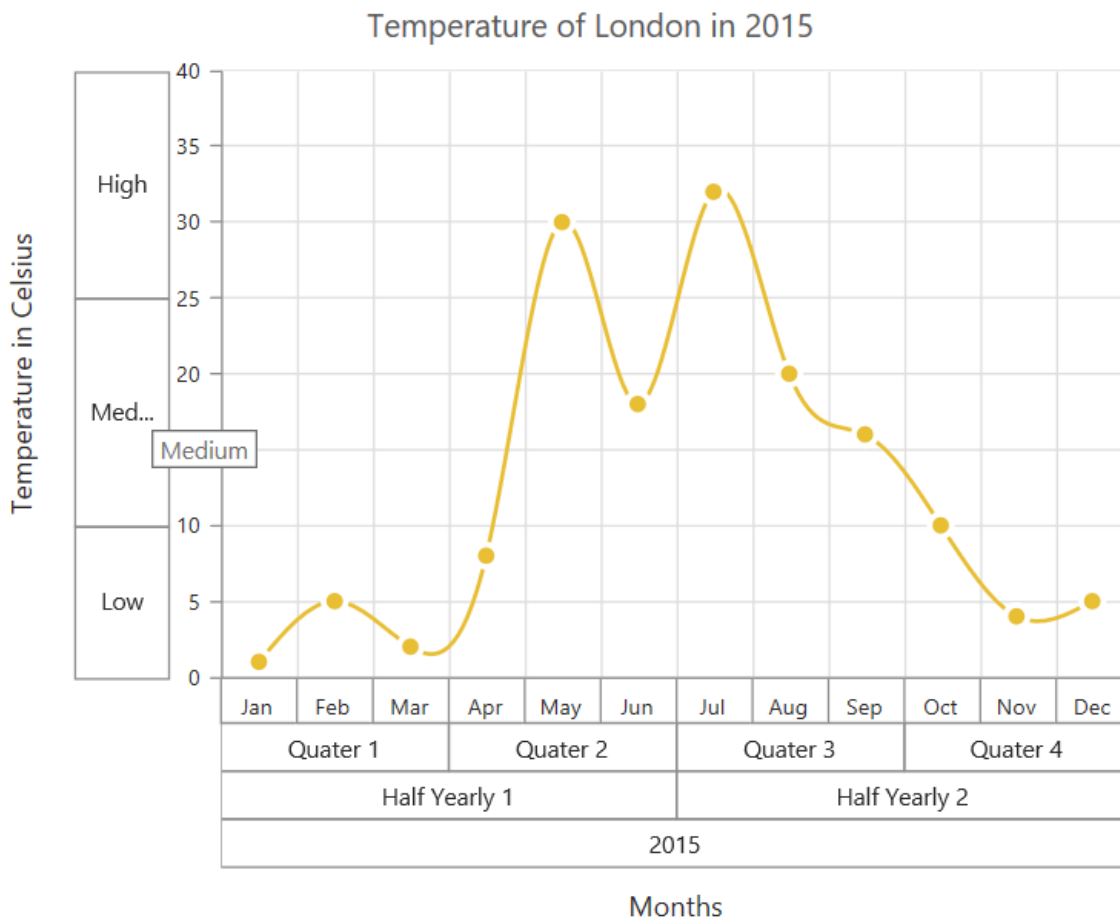


You can trim, wrap or wrapAndTrim the text if it exceeds the maximum text width value using the property `textOverflow`

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  {
    primaryXAxis:
    {
      multiLevelLabels: [
        {
          // customizing the text overflow
          textOverflow: "trim",
          maximumTextWidth: 40
        }
      ]
    }
  }
});
```

The below screenshot shows the trimmed multi-level labels



And these labels can be placed in various rows using the `[level]` property.

### Multiple panes

Chart area can be divided into multiple panes using the [rowDefinitions](#) and [columnDefinitions](#) properties.

#### Row Definitions

To split the chart area vertically into a number of rows, use [rowDefinitions](#) of the chart.

- You can allocate space for each row by using the [unit](#) option that determines whether the chart area should be split by *percentage* or *pixels* for the given [rowHeight](#) value of the [rowDefinitions](#).
- To associate a vertical axis to a row, specify the [rowDefinitions](#) **index** value to the [rowIndex](#) property of the chart axis.
- To customize each row's horizontal line, use [lineColor](#) and [lineWidth](#) property.

### JAVASCRIPT

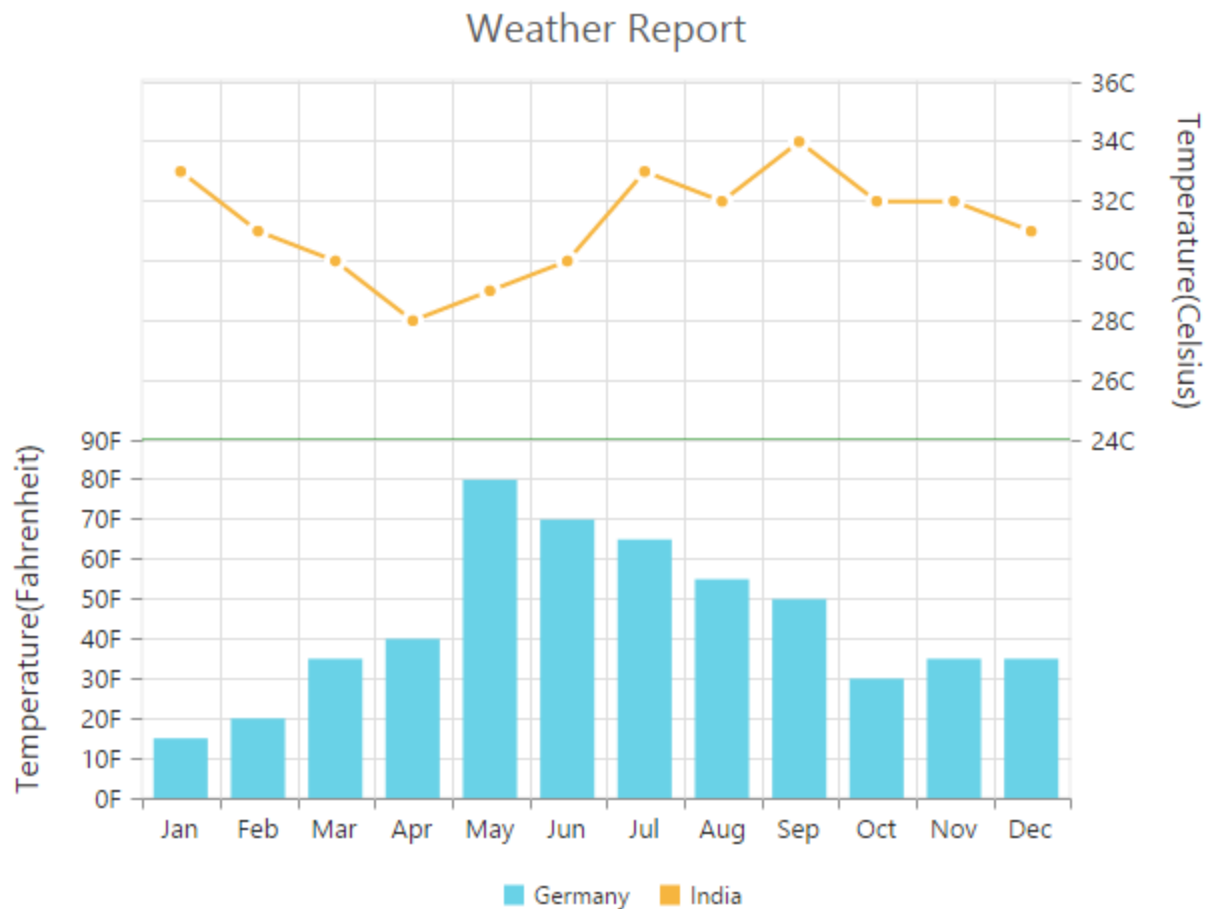
```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
$(function () {
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {

```



```
// Splitting chart area into multiple rows
rowDefinitions: [{
  // Split first row of the chart area
  unit : 'percentage',
  lineColor : 'Gray',
  rowHeight : 50,
  linewidth : 0
}, {
  // Split second row of the chart area
  unit : 'percentage',
  lineColor : 'green',
  rowHeight : 50,
  linewidth : 0
}],
axes: [{
  //Create secondary axis and bind it to second row of chart area
  name: "yAxis1",
  rowIndex: 1
}],
series: [{
  //Binding vertical axis name
  yAxisName: "yAxis1",
  // ...
}],
// ...
});
});
}
```

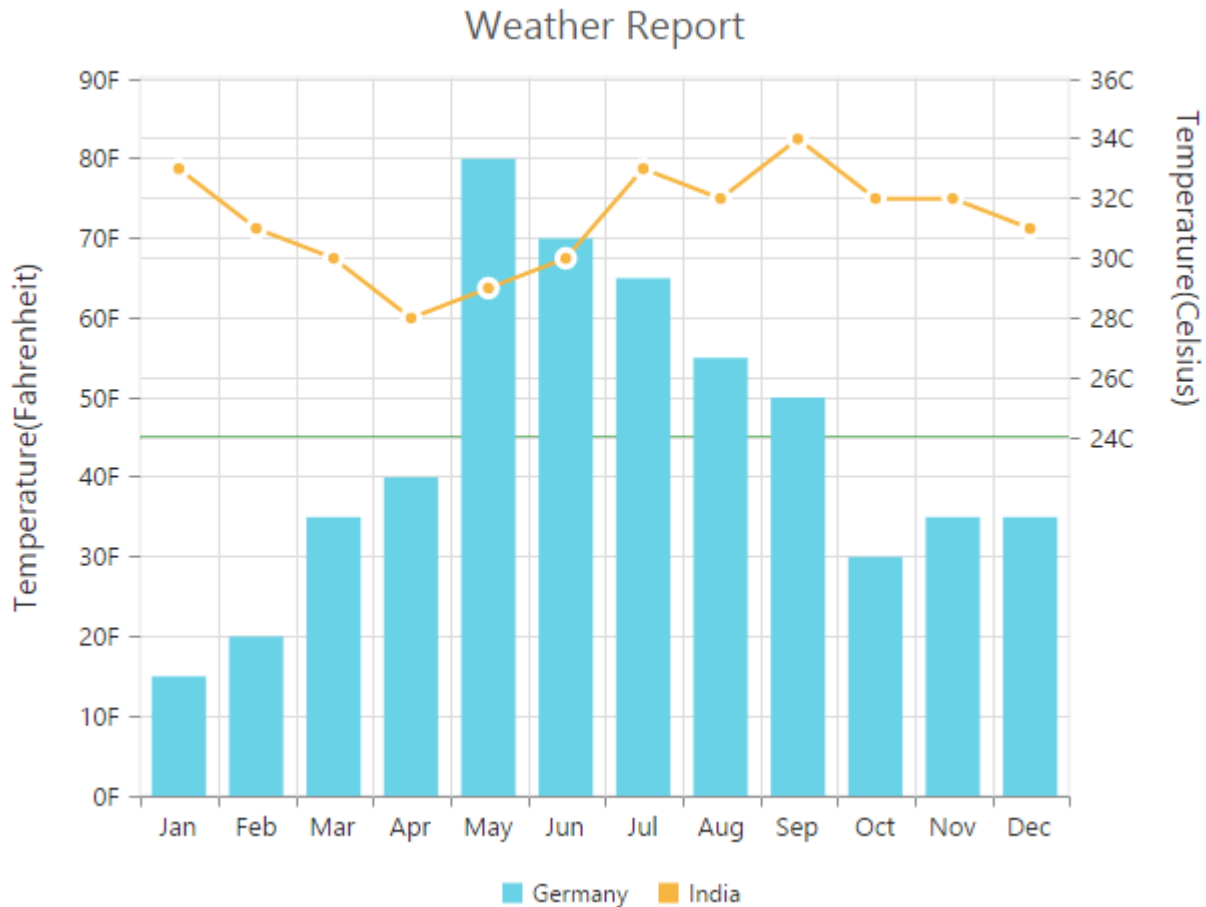


## Row Span

For spanning the vertical axis along multiple panes vertically, you can use [rowSpan](#) property of axis.

## JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  rowDefinitions: [{
    // ...
  }, {
    // ...
  }],
  axes: [{
    // ...
  }],
  primaryYAxis: {
    // Span the PrimaryYAxis
    rowspan : 2,
  },
  series: [{
    // ...
  }, {
    // ...
  }];
});
```



#### Column Definitions

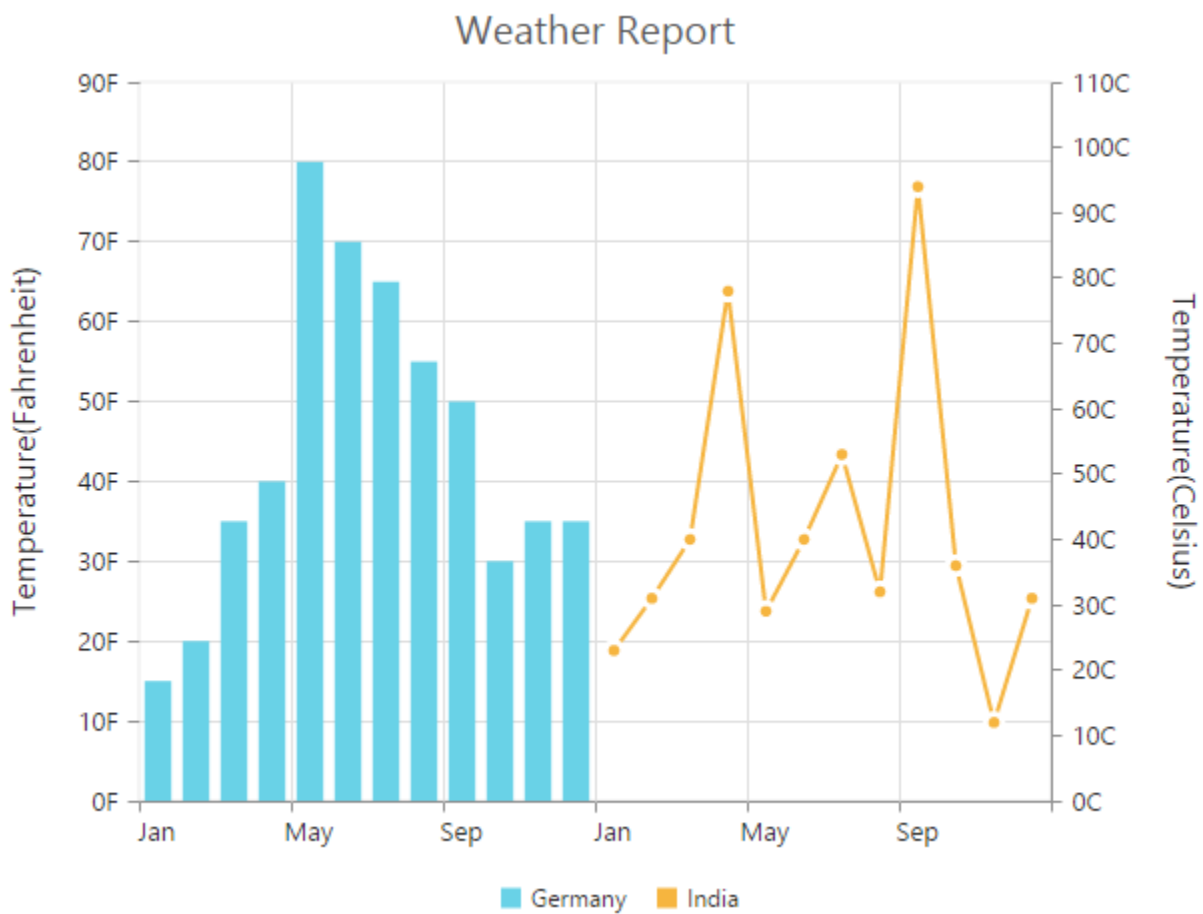
To split the chart area horizontally into a number of columns, use [columnDefinitions](#) of the chart.

- You can allocate space for each column by using the [unit](#) option that determines whether the chart area should be split by *percentage* or *pixels* for the given [columnWidth](#) value of the [columnDefinitions](#).
- To associate a horizontal axis to a column, specify the [columnDefinitions](#) **index** value to the [columnIndex](#) property of the chart axis.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // Splitting chart area into multiple columns
  columnDefinitions: [{
    // Split first column of the chart area
    unit : 'percentage',
    columnWidth : 50,
  }, {
    // Split second column of the chart area
    unit : 'percentage',
    columnWidth : 50,
  }],
  axes: [{
```

```
//Create secondary axis and bind it to second column of chart area
name: "xAxis1",
columnIndex: 1
}],
series: [{
//Binding horizontal axis name
xAxisName: "xAxis1",
// ...
}],
// ...
});
```



### Column Span

For spanning the horizontal axis along multiple panes horizontally, you can use [columnSpan](#) property of axis.

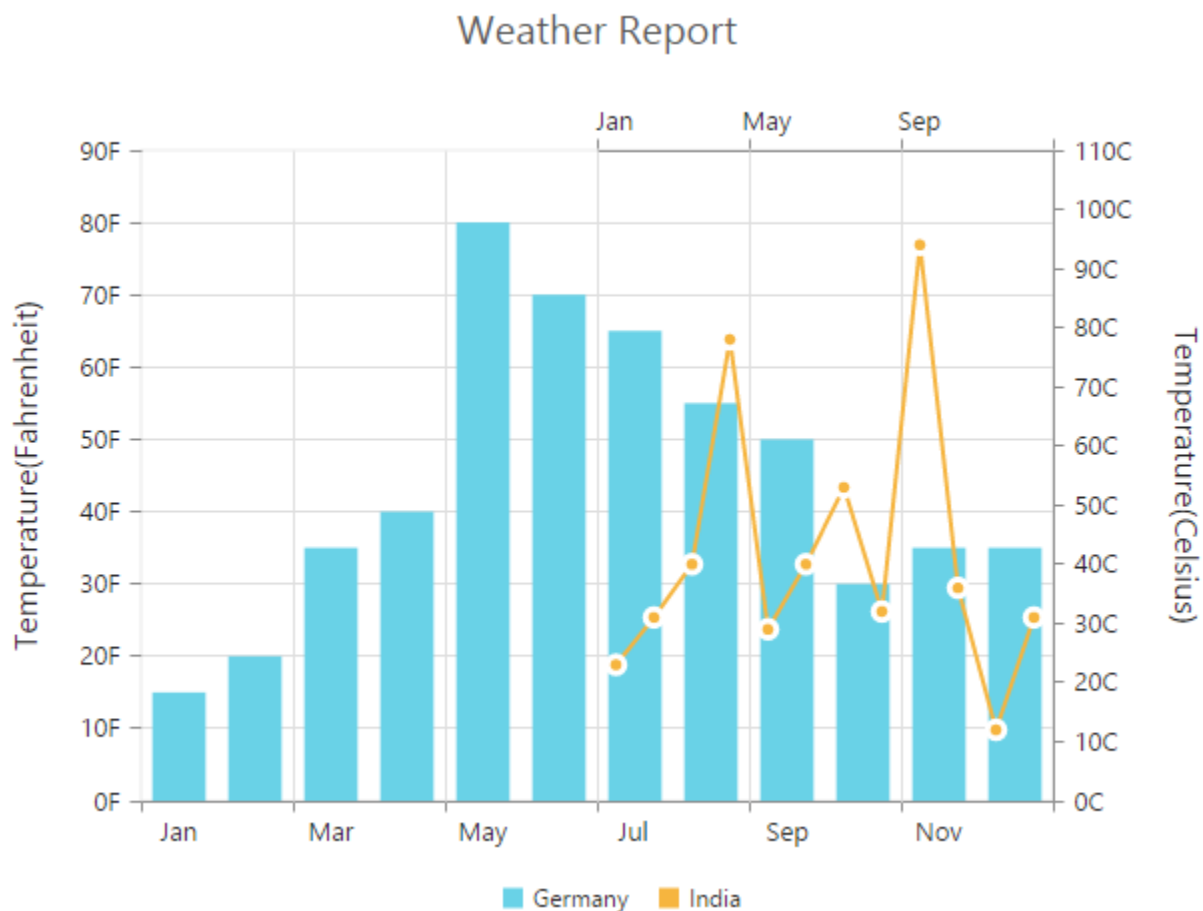
### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
columnDefinitions: [{
// ...
}], {
// ...
});
```

```

    }],
    axes: [{
      // ...
    }],
    primaryXAxis: {
      // Span the PrimaryXAxis
      columnSpan : 2,
    },
    series: [{
      // ...
    }],
    // ...
  });

```



## ChartTypes

### Line Chart

To render a Line Chart, set the series [type](#) as “**line**” in the chart series. To change the line segment color, you can use the [fill](#) property of the series.

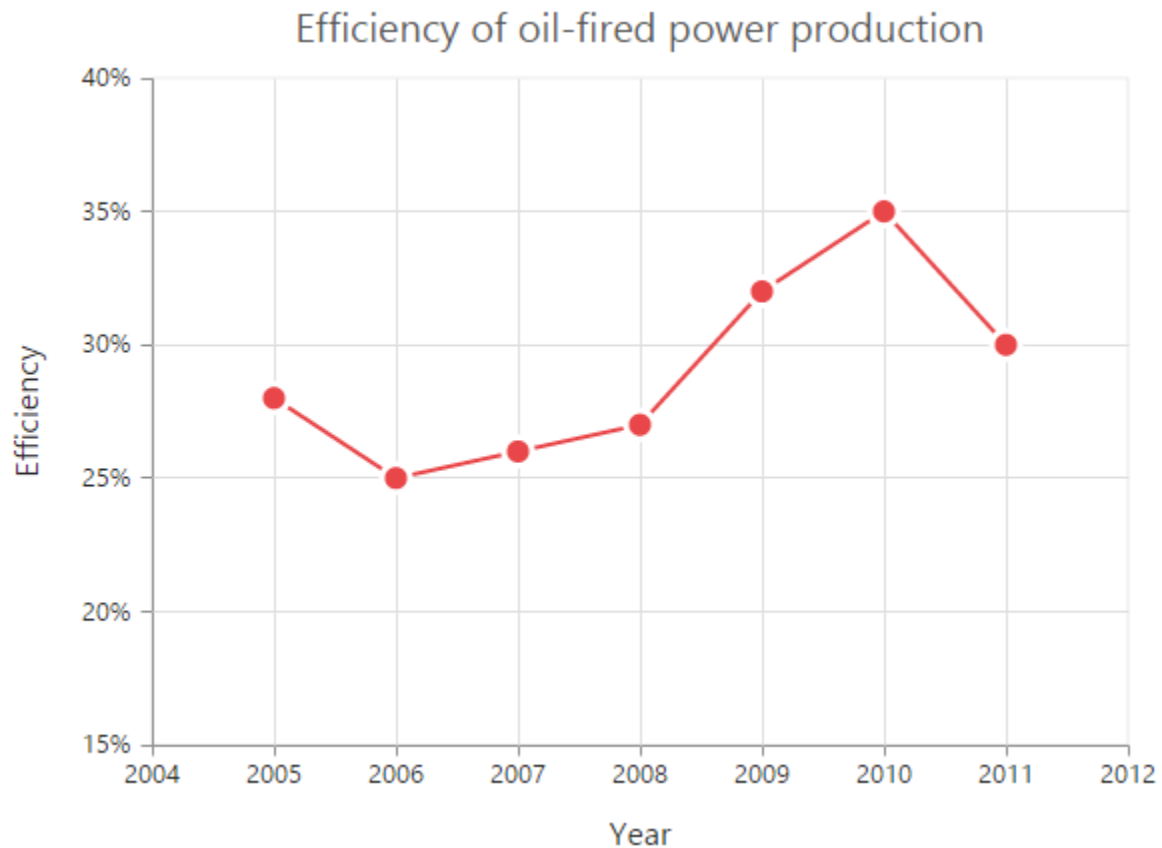
### JAVASCRIPT

```

var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{

```

```
//Change type and color of the series.  
type: 'line',  
fill: "#E94649",  
// ...  
}],  
// ...  
});
```

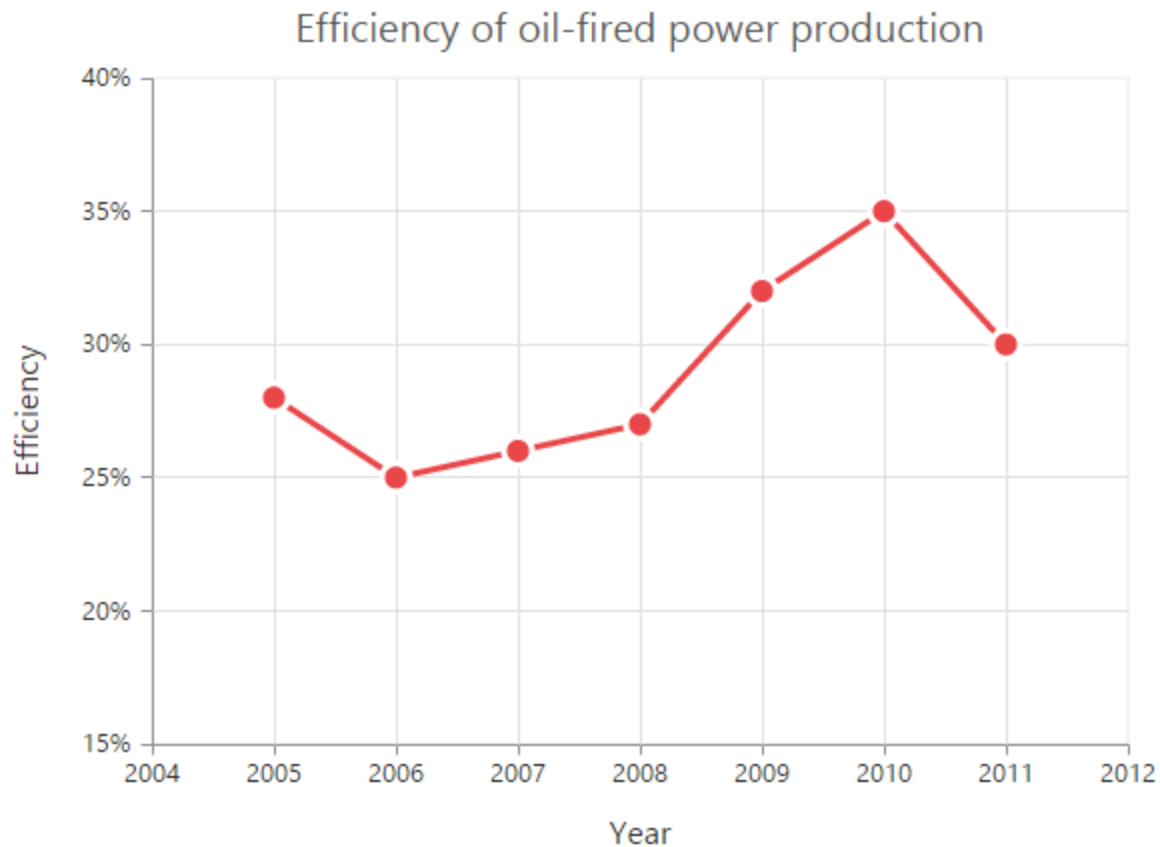


#### Change the line width

To change the width of the line segment, you can use the [width](#) property in the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  //...  
  series: [{  
    //Change the width of line series  
    width: 3,  
    // ...  
  }],  
  //...  
});
```

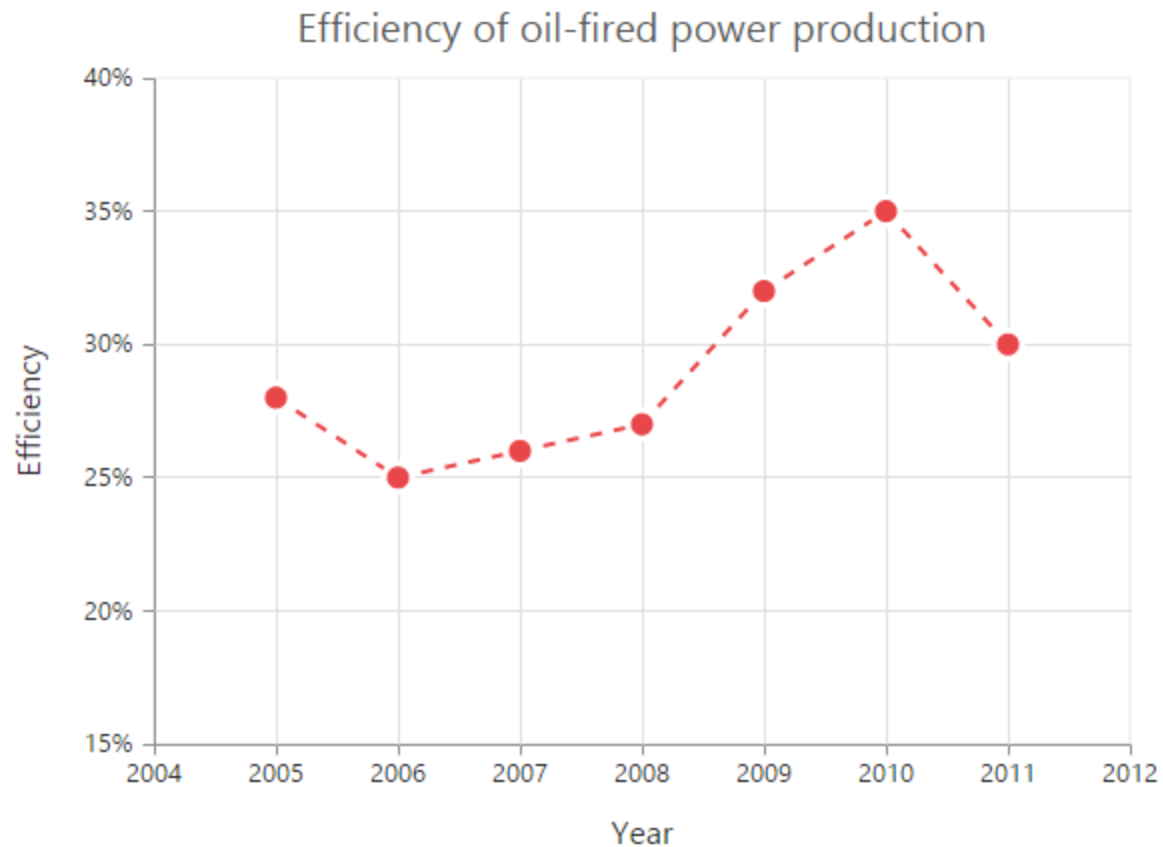


#### Dashed lines

To render the line series with dotted lines, you can use the [dashArray](#) option of the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change dash array to display dotted or dashed lines  
    dashArray: '5,5',  
    // ...  
  }],  
  // ...  
});
```



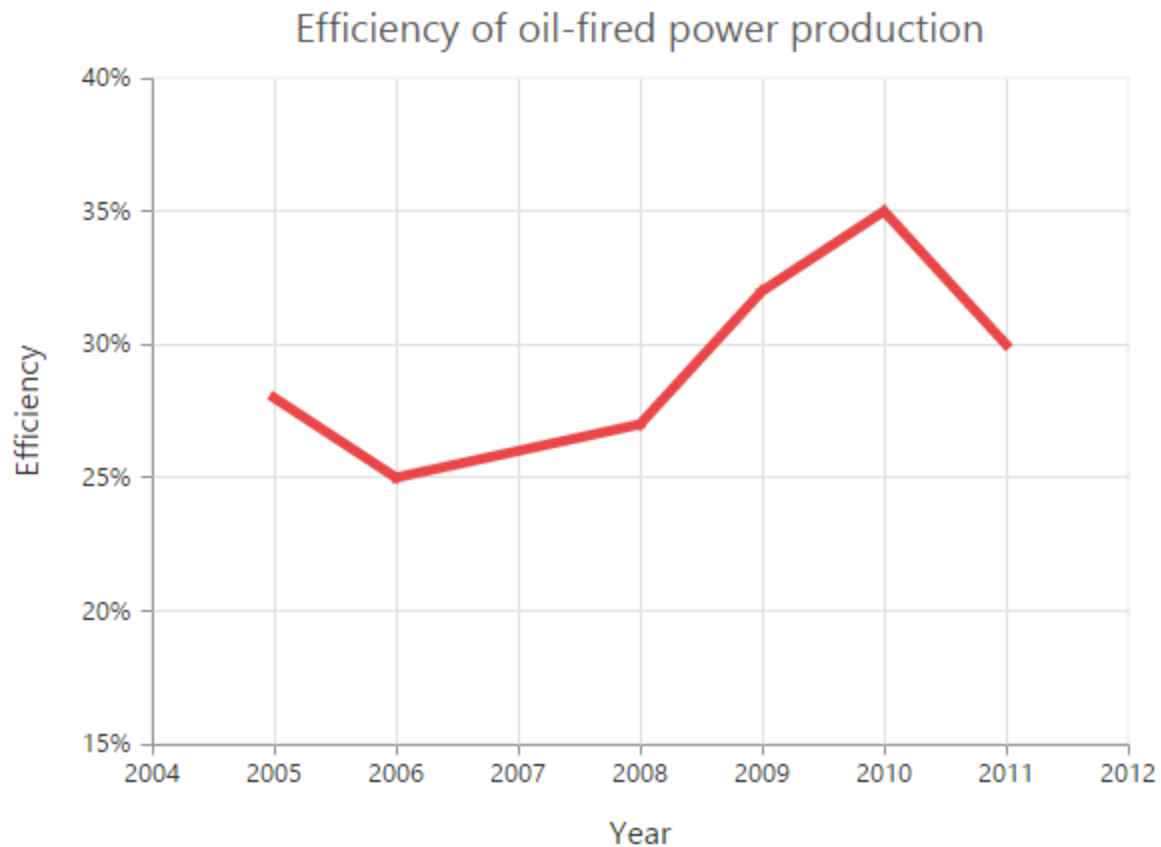
#### Changing the line cap

For customizing the start and end caps of the line segment, you can use the [lineCap](#) property.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change line cap  
    lineCap: 'square',  
    // ...  
  }],  
  // ...  
});
```



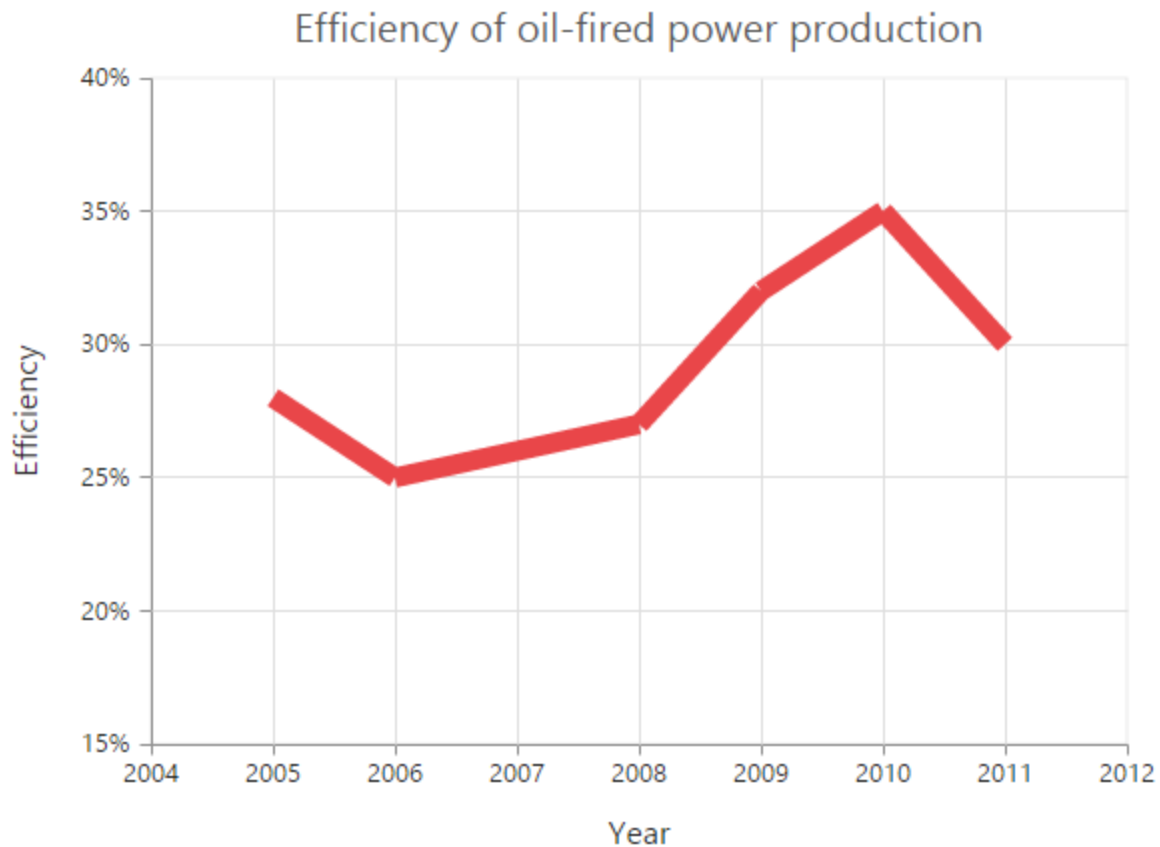


#### Changing the line join

You can use the [lineJoin](#) property to specify how two intersecting line segments should be joined.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  //...  
  series: [{  
    //Change line join  
    lineJoin: 'round',  
    //...  
  }],  
  //...  
});
```

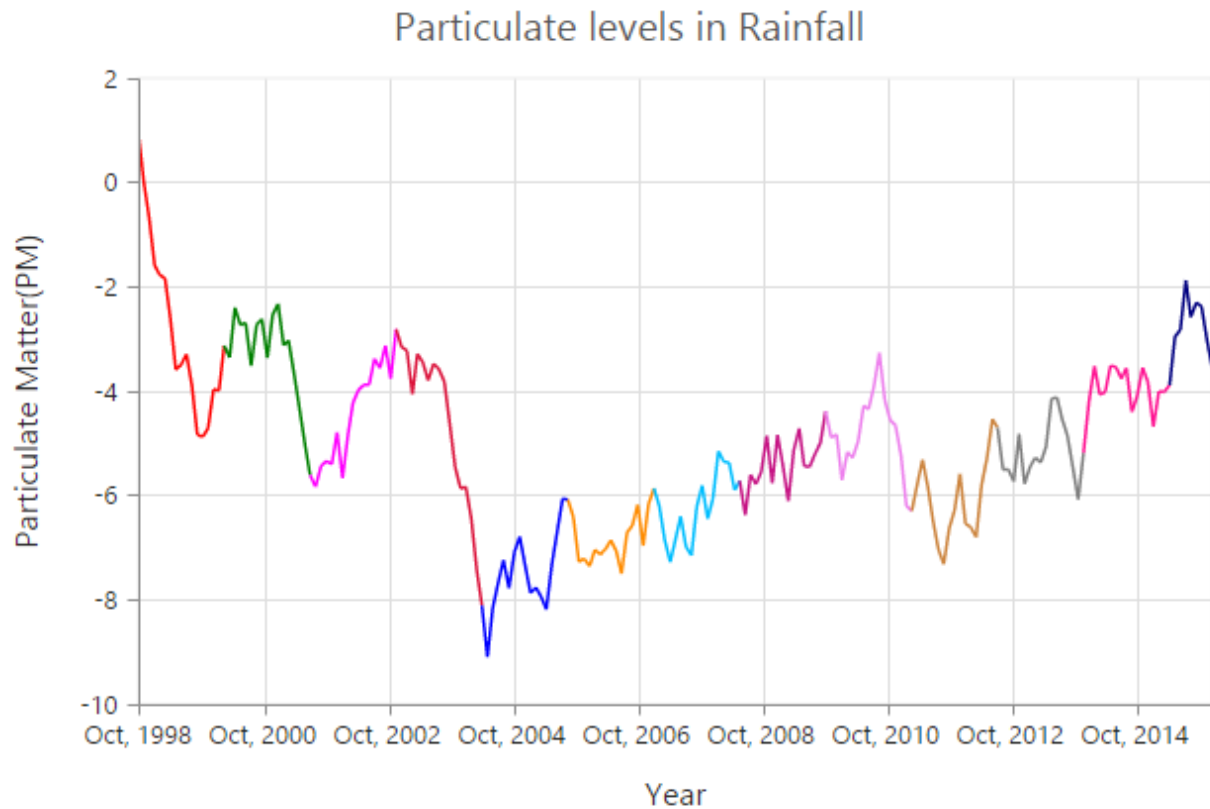


#### MultiColor Line

You can change the color of the line segments by using the [fill](#) property of the each [points](#) in the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    // Change the color of a line  
    points: [{ fill: 'red' },  
    // ...  
  ],  
  // ...  
}],  
  // ...  
});
```

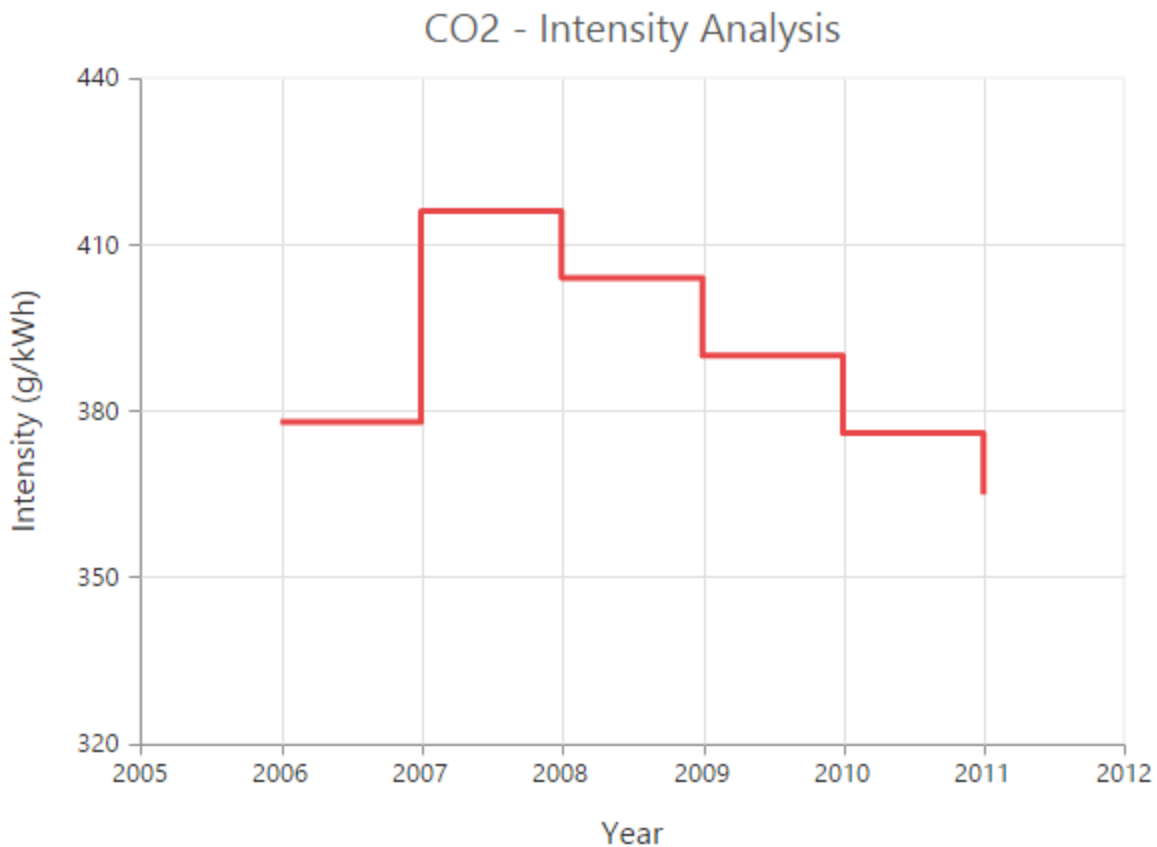


### Step Line Chart

To render a Step Line Chart, set the series [type](#) as “**stepline**” in the chart series. To change the StepLine segment color, you can use the [fill](#) property of the series.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change type and color of the series.  
    type: 'line',  
    fill: "#E94649",  
    // ...  
  }],  
  // ...  
});
```

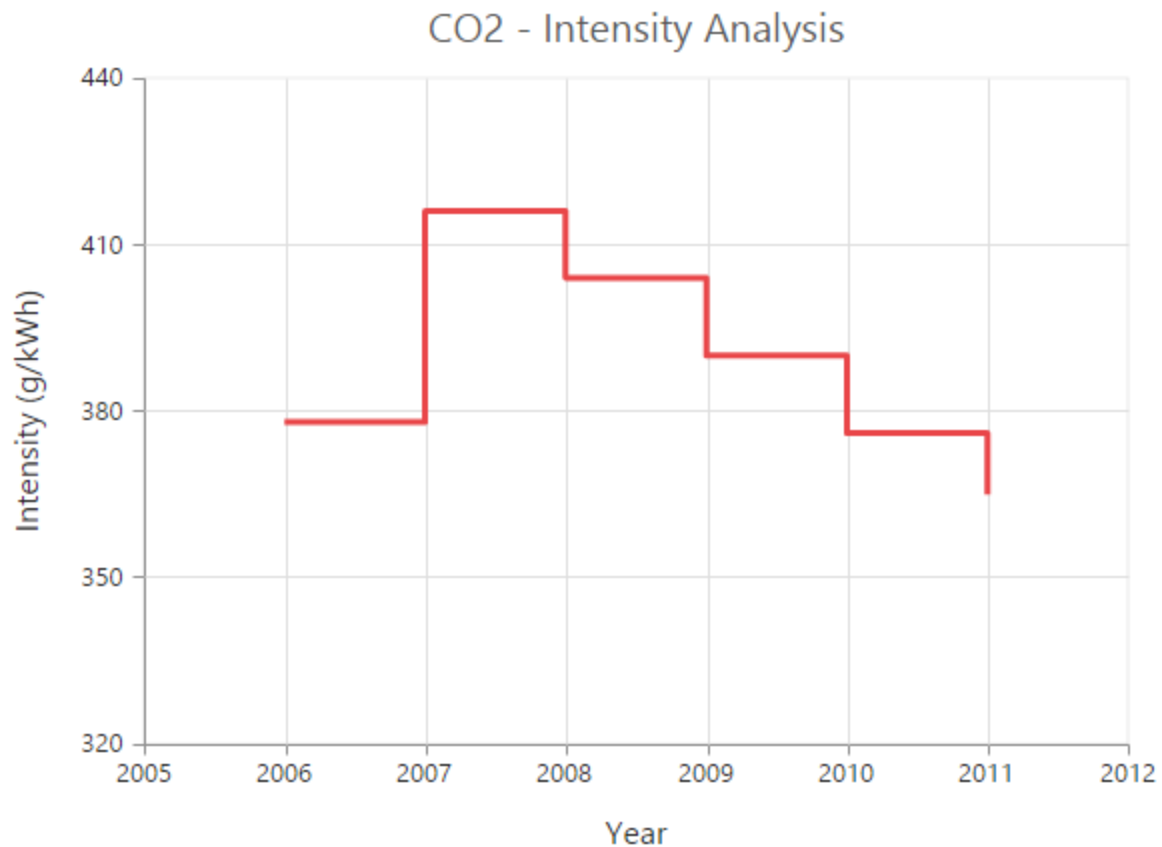


#### Changing the line width

To change the line width, you can use the **width** property.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  //...  
  series: [{  
    //Change the width of step line series  
    width: 3,  
    // ...  
  }],  
  //...  
});
```

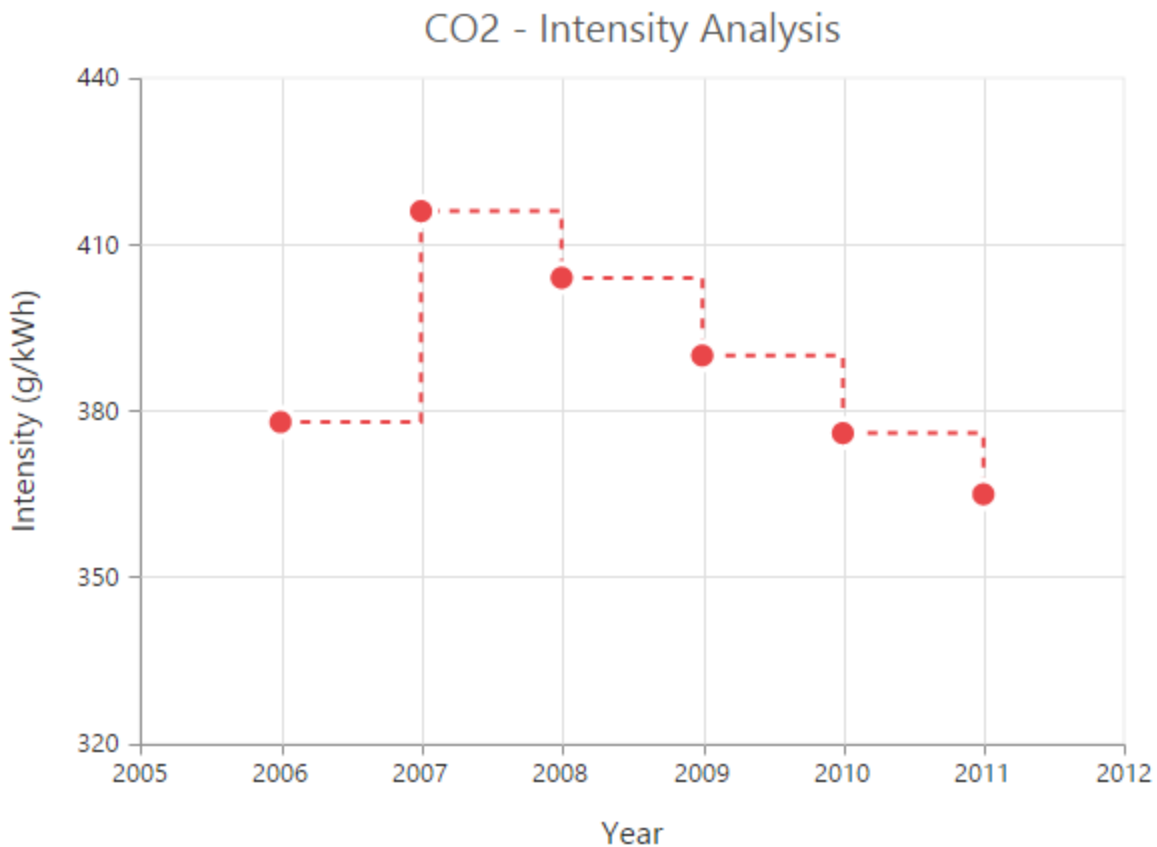


#### Dashed lines

To render the step line series with dotted lines, you can use the [dashArray](#) option of the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change dash array to display dotted or dashed lines  
    dashArray: '5,5',  
    // ...  
  }],  
  // ...  
});
```

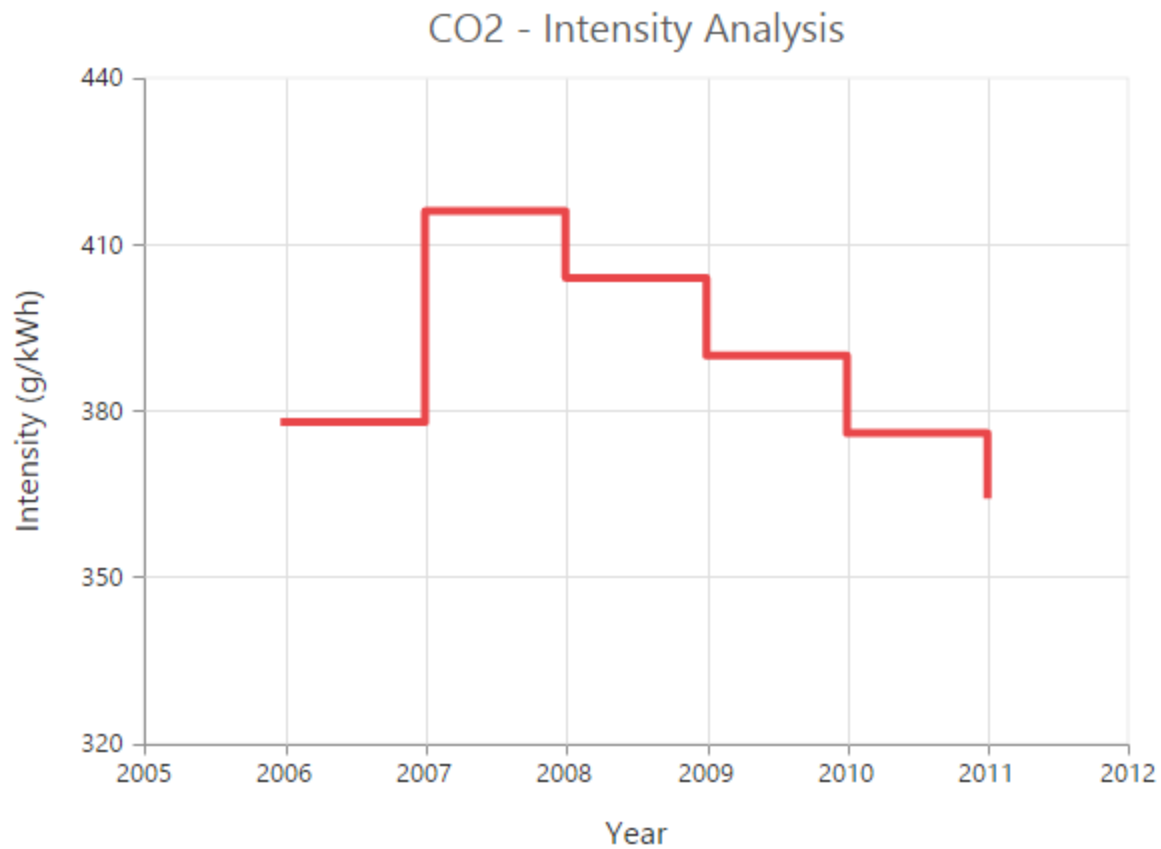


#### Changing the line cap

For customizing the start and end caps of the line segment, you can use the [lineCap](#) property.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change line cap  
    lineCap: 'square',  
    // ...  
  }],  
  // ...  
});
```

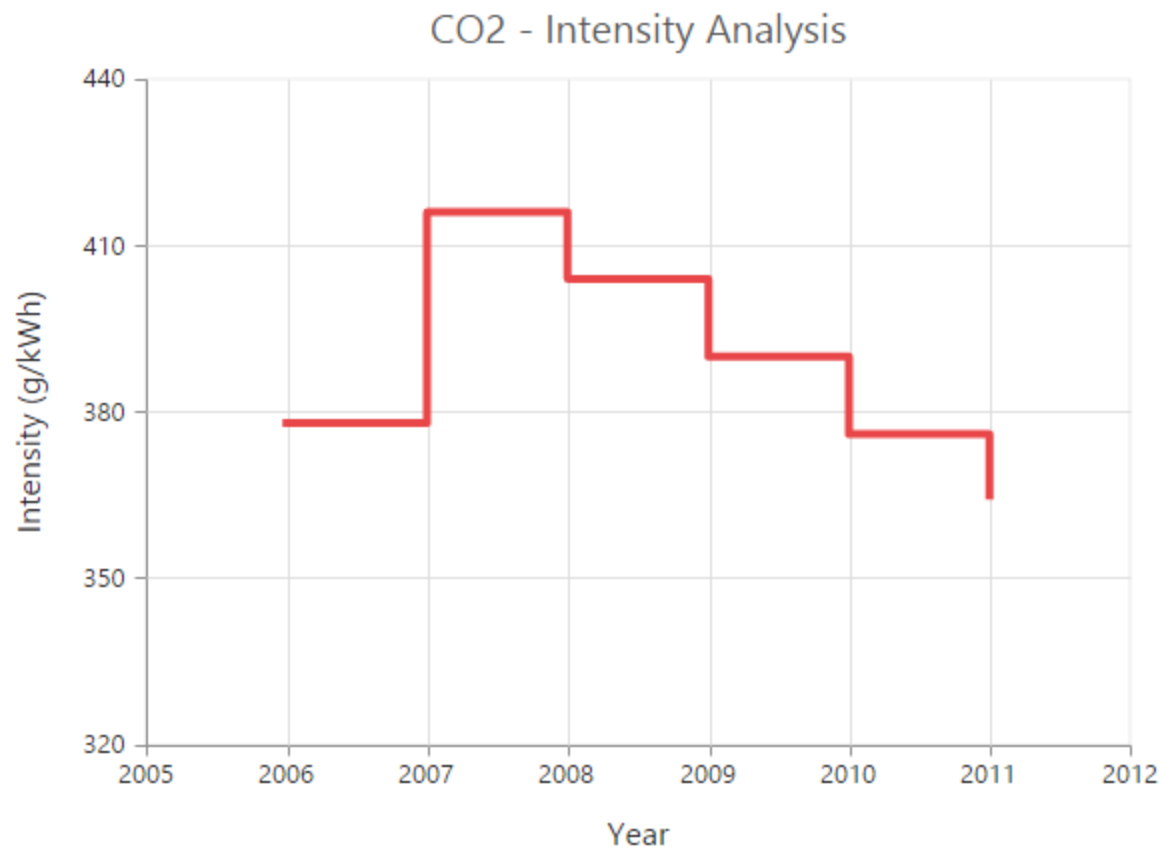


#### Changing the line join

You can use the [lineJoin](#) property to specify how two intersecting line segments should be joined.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  //...  
  series: [{  
    //Change line join  
    lineJoin: 'round',  
    //...  
  }],  
  //...  
});
```



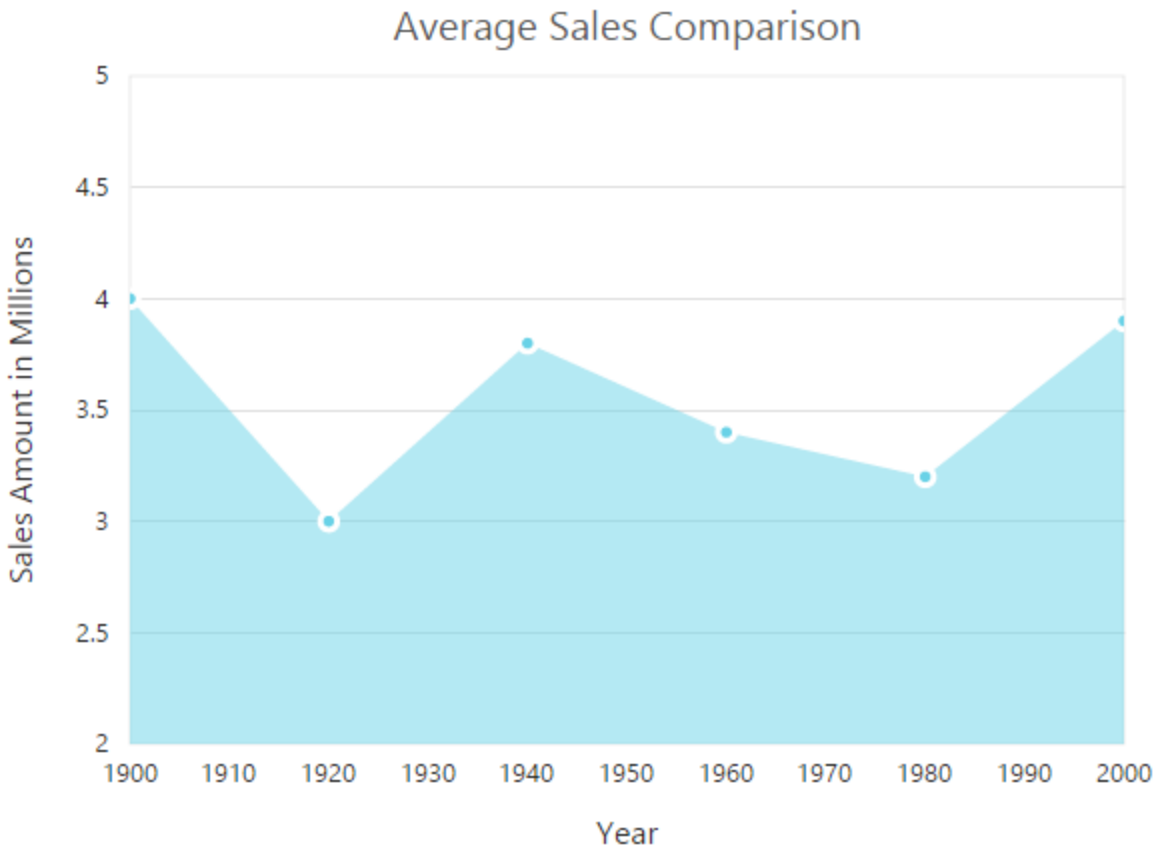
#### Area Chart

To render an Area chart, you can specify the series [type](#) as “area” in the chart series. To change the Area color, you can use the [fill](#) property of the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    // Change the series type and fill color  
    type: 'area',  
    fill: '#69D2E7'  
    // ...  
  }],  
  // ...  
});
```





#### Range Area Chart

To render a Range Area Chart, set the [type](#) as “**rangeArea**” in the chart series. To change the RangeArea color, you can use the [fill](#) property of the series.

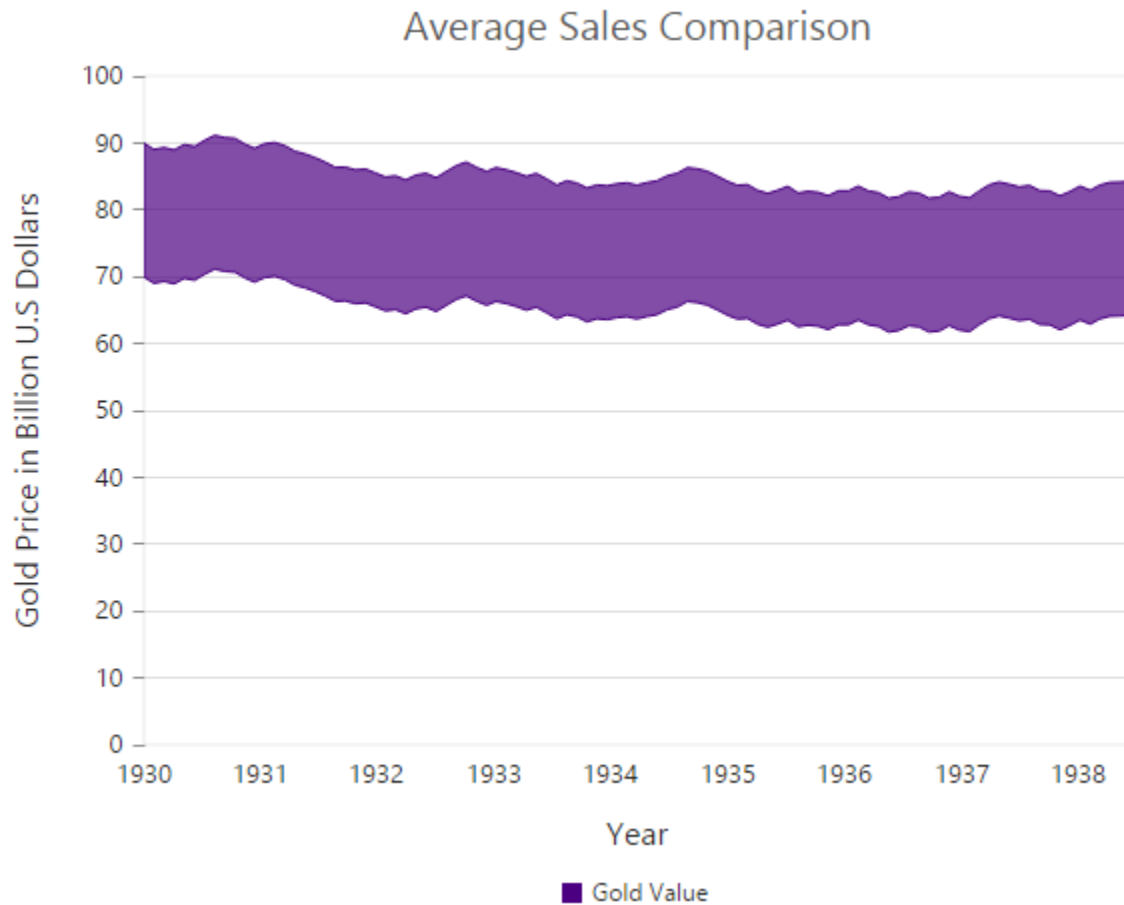
Since the RangeArea series requires two y values for a point, you have to add the [high](#) and [low](#) value. High and Low value specifies the maximum and minimum range of the points.

- When you are using the [points](#) option, specify the high and low values by using the [high](#) and [low](#) option of the point.
- When you are using the [dataSource](#) option to assign the data, map the fields from the dataSource that contain high and low values by using the [series.high](#) and [series.low](#) options.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Change the series type and fill color
    type: 'rangeArea',
    fill: "Indigo",
    //Use high and low values instead of y
    points:[{ x: 1935, high:80, low:70 },
    // ...
    ],
    // ...
  }],
});
```

```
});
```

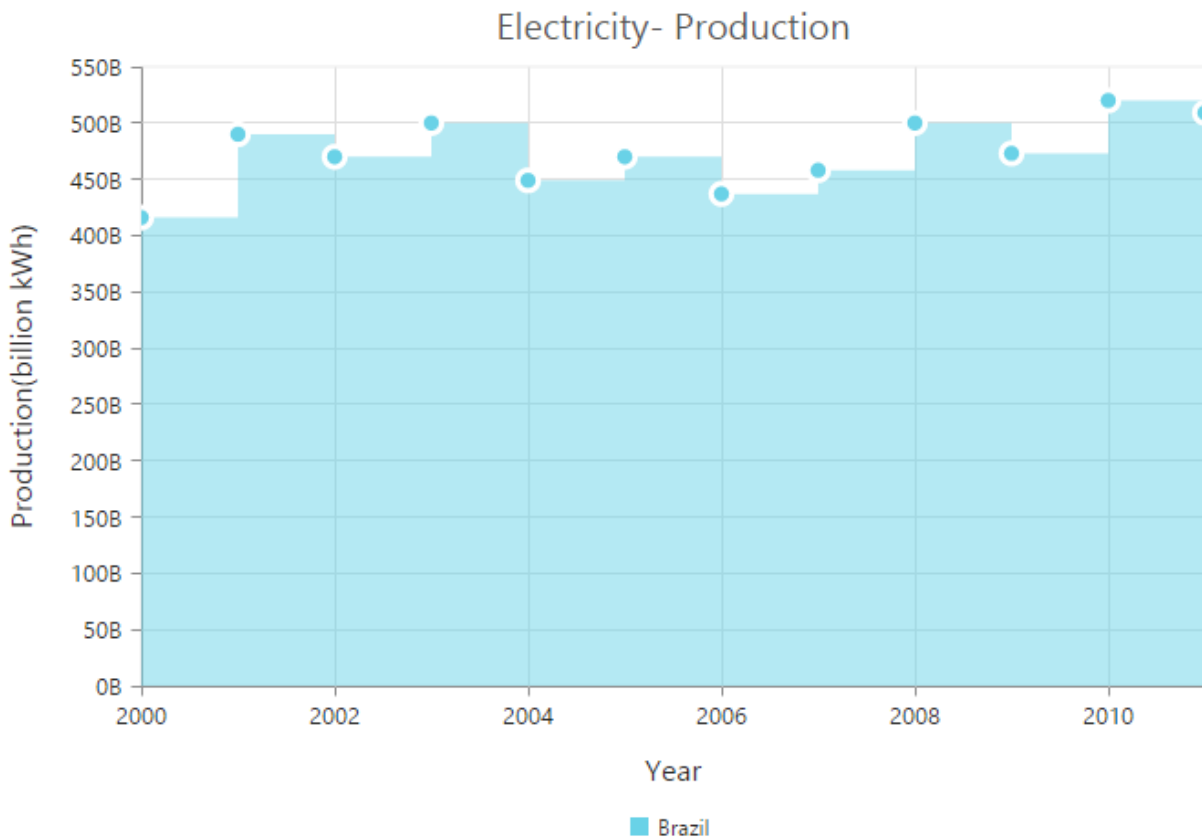


### Step Area Chart

To render a Step Area Chart, set the [type](#) as “**stepArea**” in the chart series. To change the StepArea color, you can use the [fill](#) property of the series.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    // Change the series type and fill color  
    type: 'stepArea',  
    fill: " #69D2E7",  
    // ...  
  }],  
  // ...  
});
```

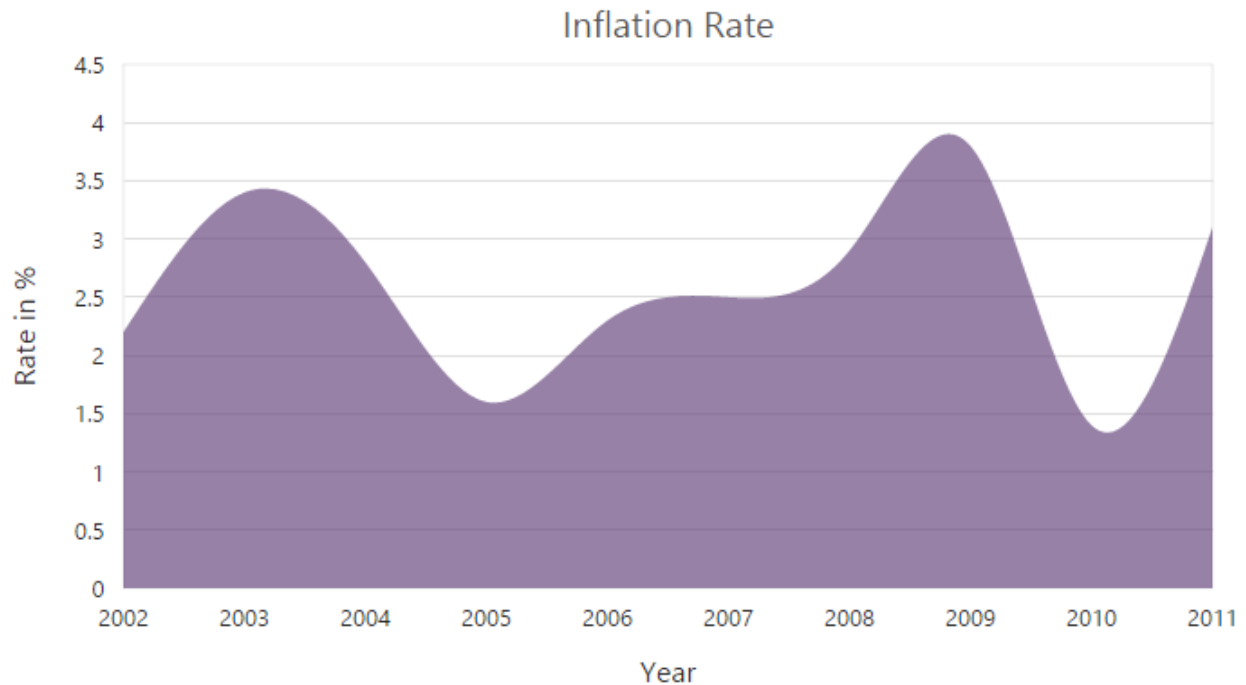


### Spline Area Chart

To render a Spline Area Chart, set the [type](#) as “**splineArea**” in the chart series. To change the SplineArea color, you can use the [fill](#) property of the series.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // Change the series type and fill color
    type: 'splineArea',
    fill: "#C4C24A",
    // ...
  }],
  //...
});
```

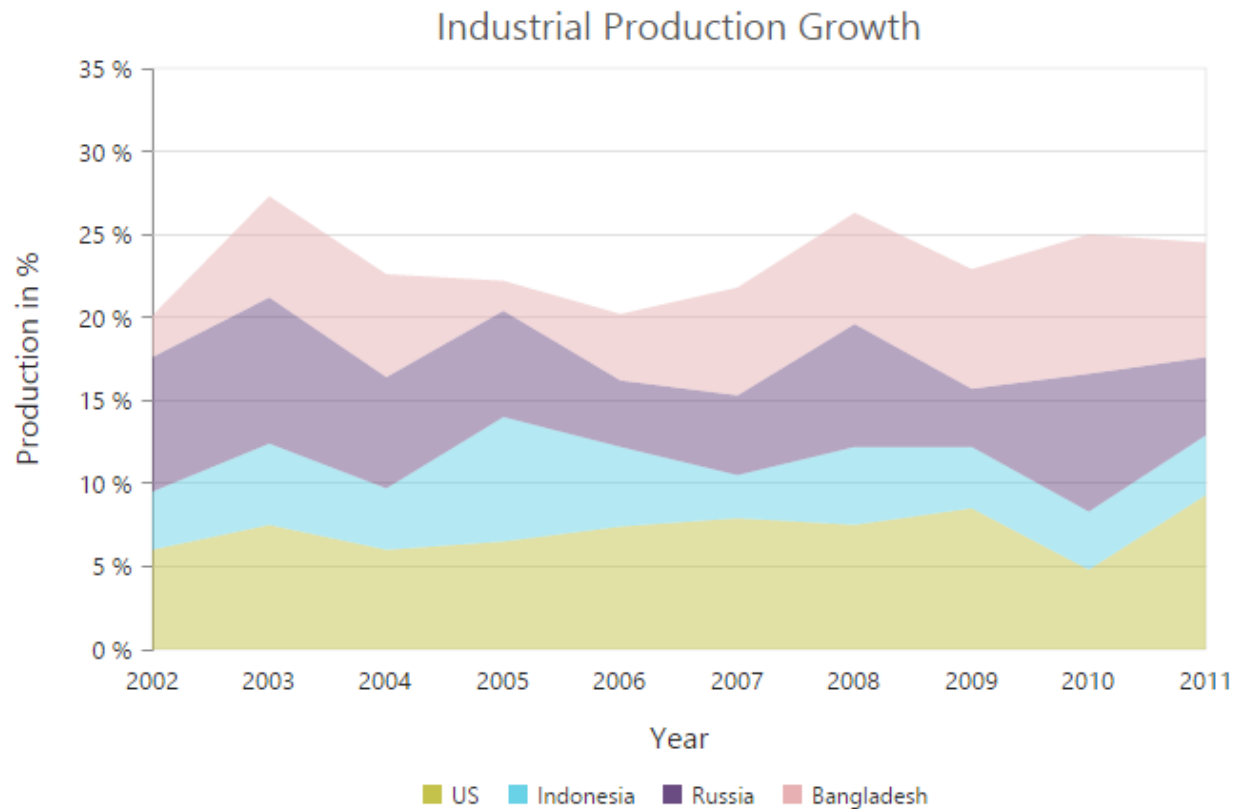


### Stacked Area Chart

To render a Stacked Area Chart, set the [type](#) as “**stackingArea**” in the chart series. To change the StackingArea color, you can use the [fill](#) property of the series.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change series type and fill color  
    type: 'stackingArea',  
    fill: "#69D2E7",  
    // ...  
  }],  
  // ...  
});
```

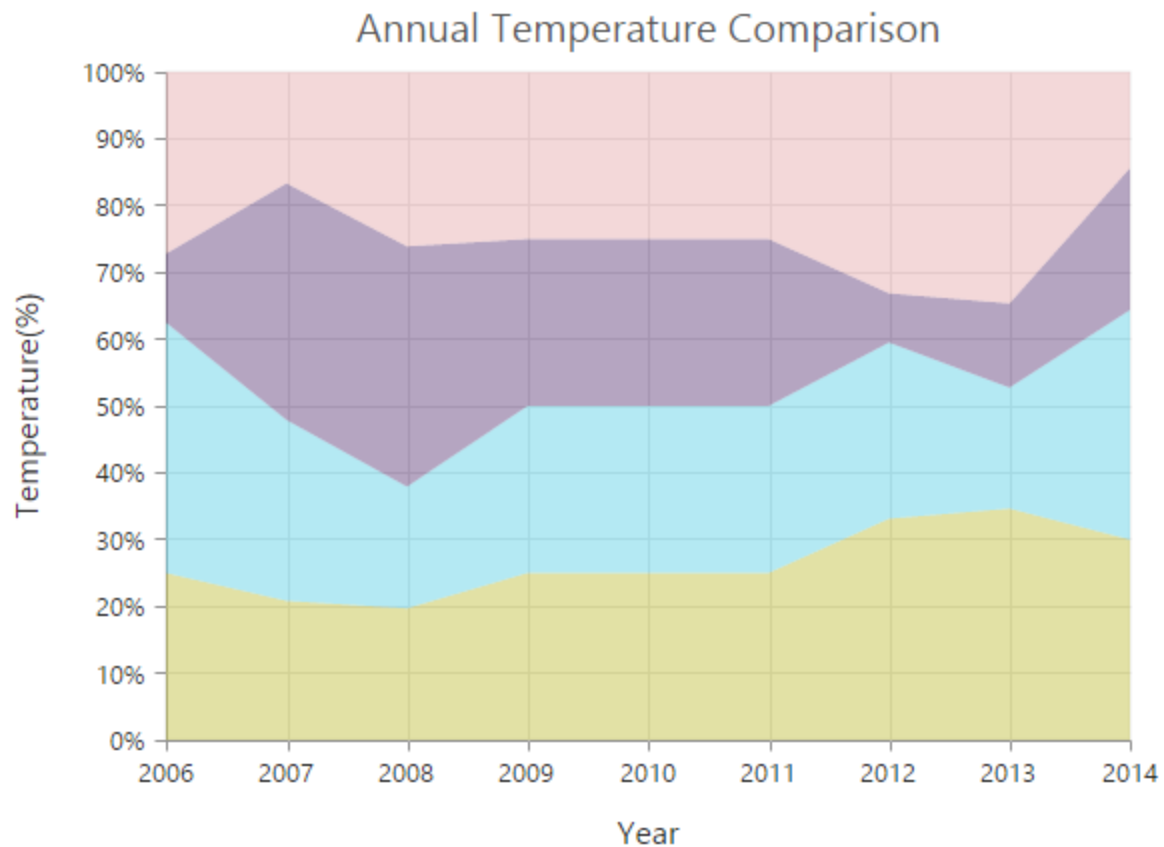


#### 100% Stacked Area Chart

To render a 100% Stacked Area Chart, set the [type](#) as “**stackingArea100**” in the chart series. To change the StackingArea100 color, you can use the [fill](#) property of the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Change type and fill color of the series
    type: 'stackingArea100',
    fill: "#C4C24A",
    // ...
  }],
  // ...
});
```

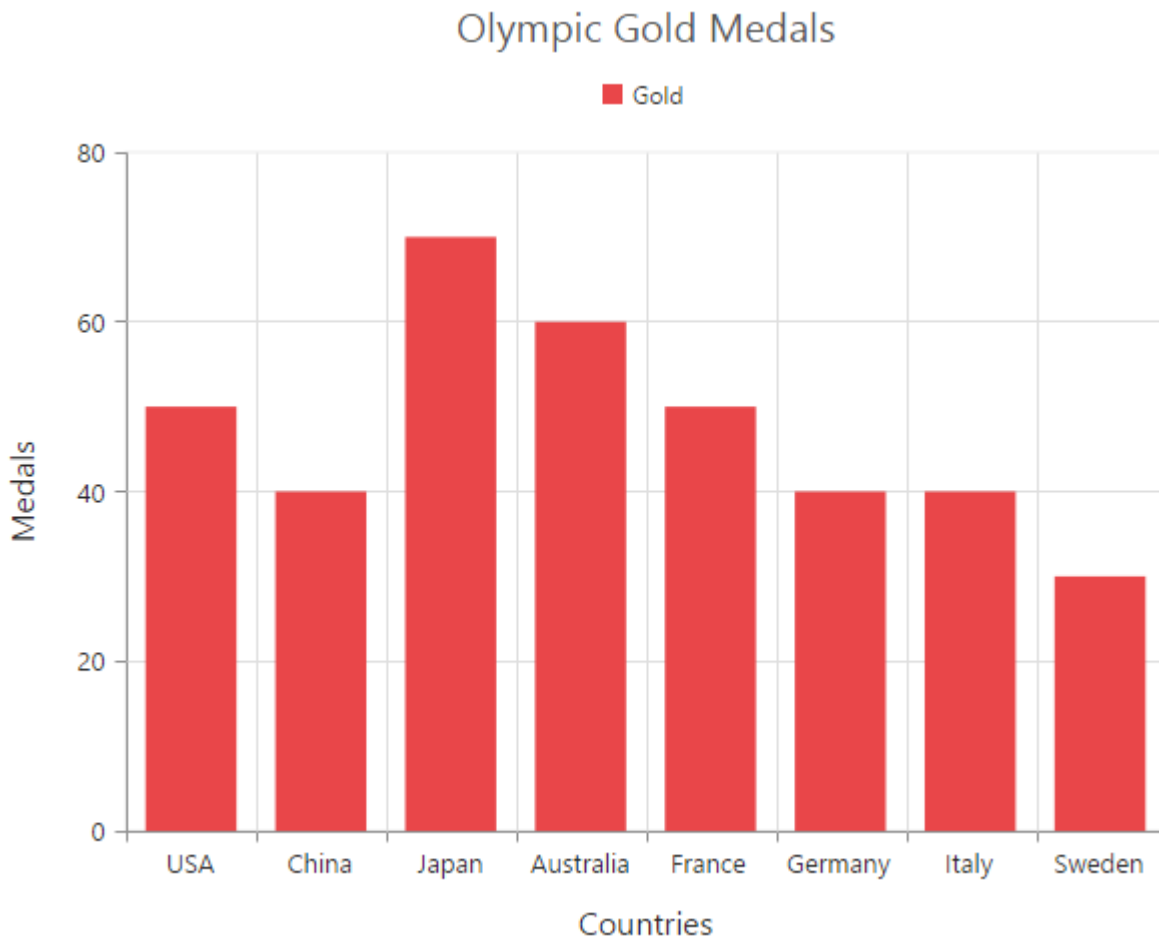


#### Column Chart

To render a Column Chart, set the [type](#) as “**column**” in the chart series. To change the color of the column series, you can use the [fill](#) property.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change type and fill color of the series  
    type: 'column',  
    fill: "#E94649",  
    // ...  
  }],  
  // ...  
});
```

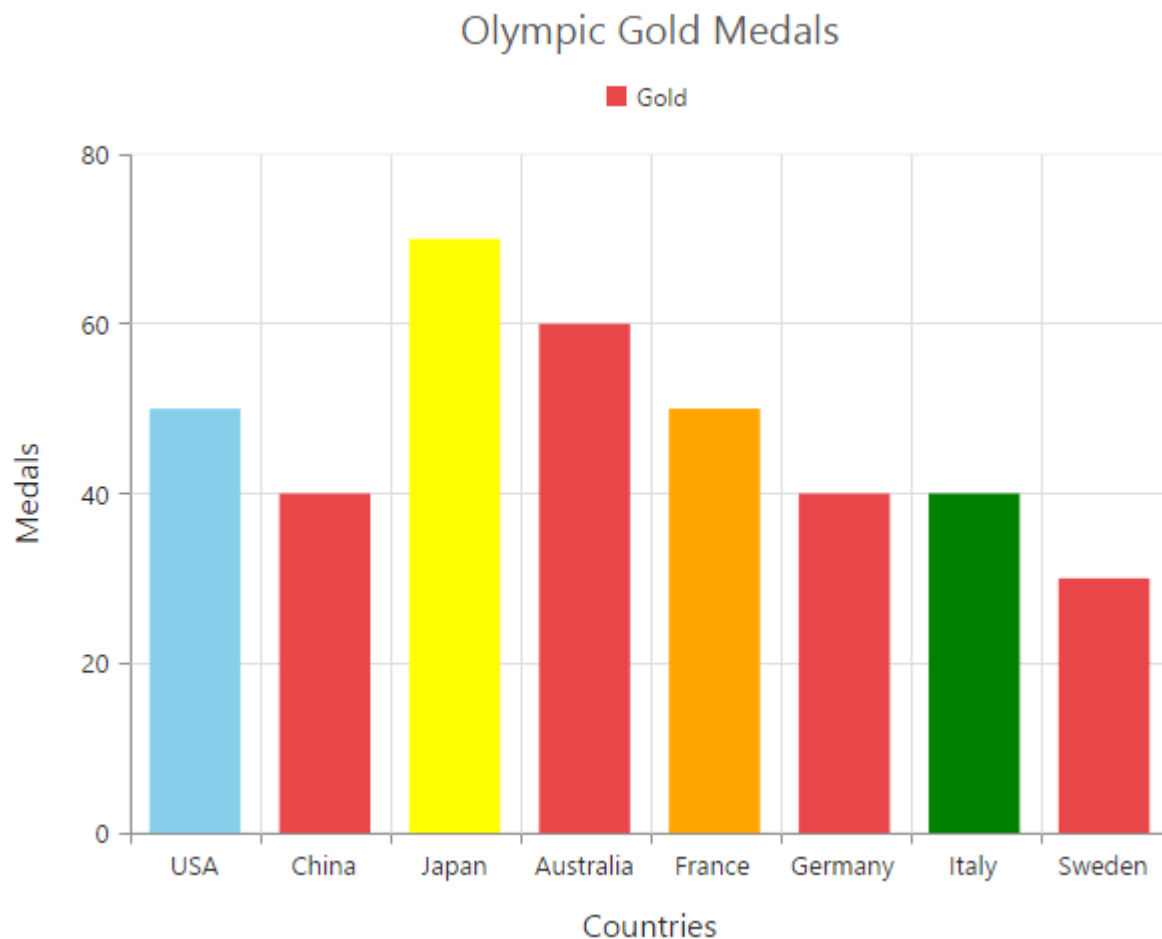


#### Change a point color

You can change the color of a column by using the [fill](#) property of the point.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    // Change the color of a column  
    points: [{ fill: 'skyblue' }],  
    // ...  
  },  
  // ...  
}],  
  // ...  
});
```



#### Column width customization

Width of the column type series can be customized by using the [columnWidth](#) property. Default value of [columnWidth](#) is 0.7. Value ranges from 0 to 1. Here 1 corresponds to 100% of available width and 0 corresponds to 0% of available width.

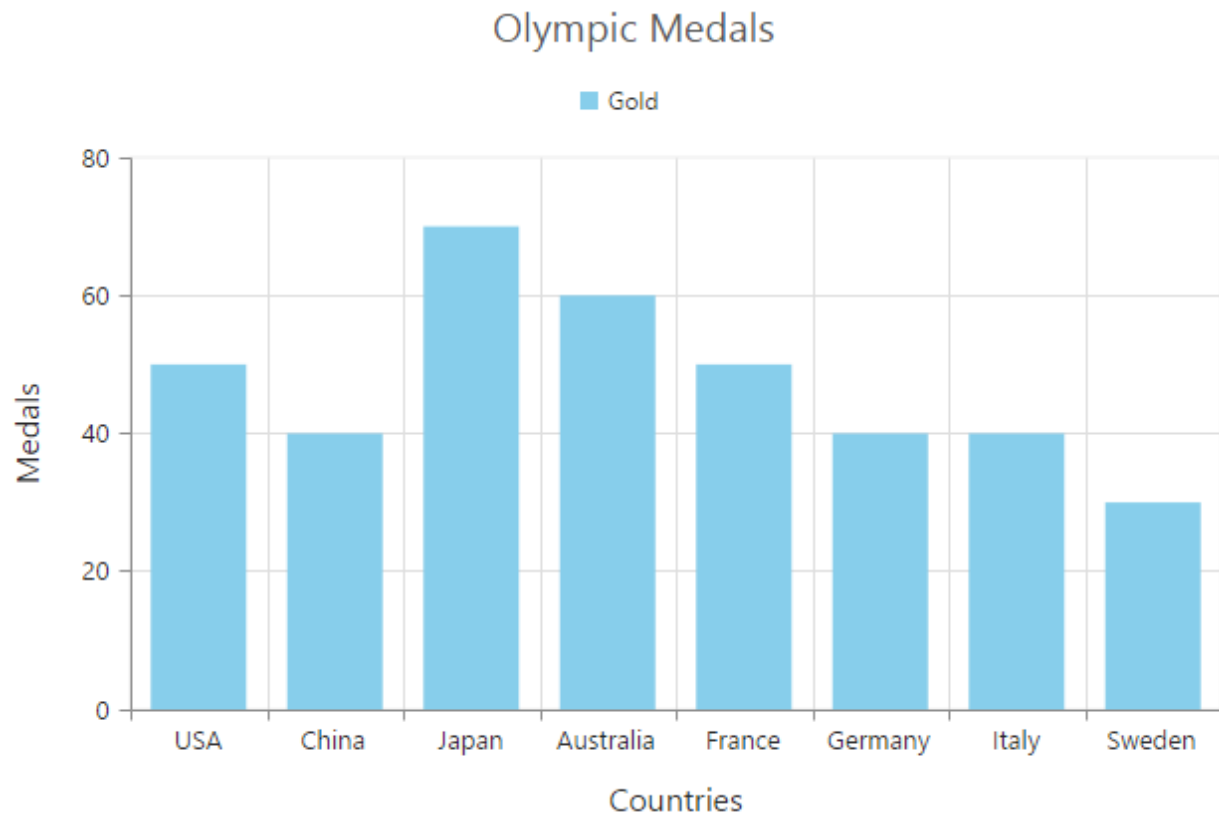
**Note:** Width of a column also depends upon the [columnSpacing](#) property, because [columnSpacing](#) will reduce the space available for drawing a column. This is also applicable for StackingColumn, StackingColumn100, Bar, StackingBar, StackingBar100, RangeColumn, HiLo, HiLoOpenClose, Candle and Waterfall charts.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  //Common settings for all series
  commonSeriesOptions: {
    //Width of columns in column type series
    columnWidth: 0.7
    //...
  },
  //Settings specific to individual series
  series: [{
    //Width of columns in column type series
    columnWidth: 0.8
  }]
});
```



```
//...  
}],  
//...  
});
```

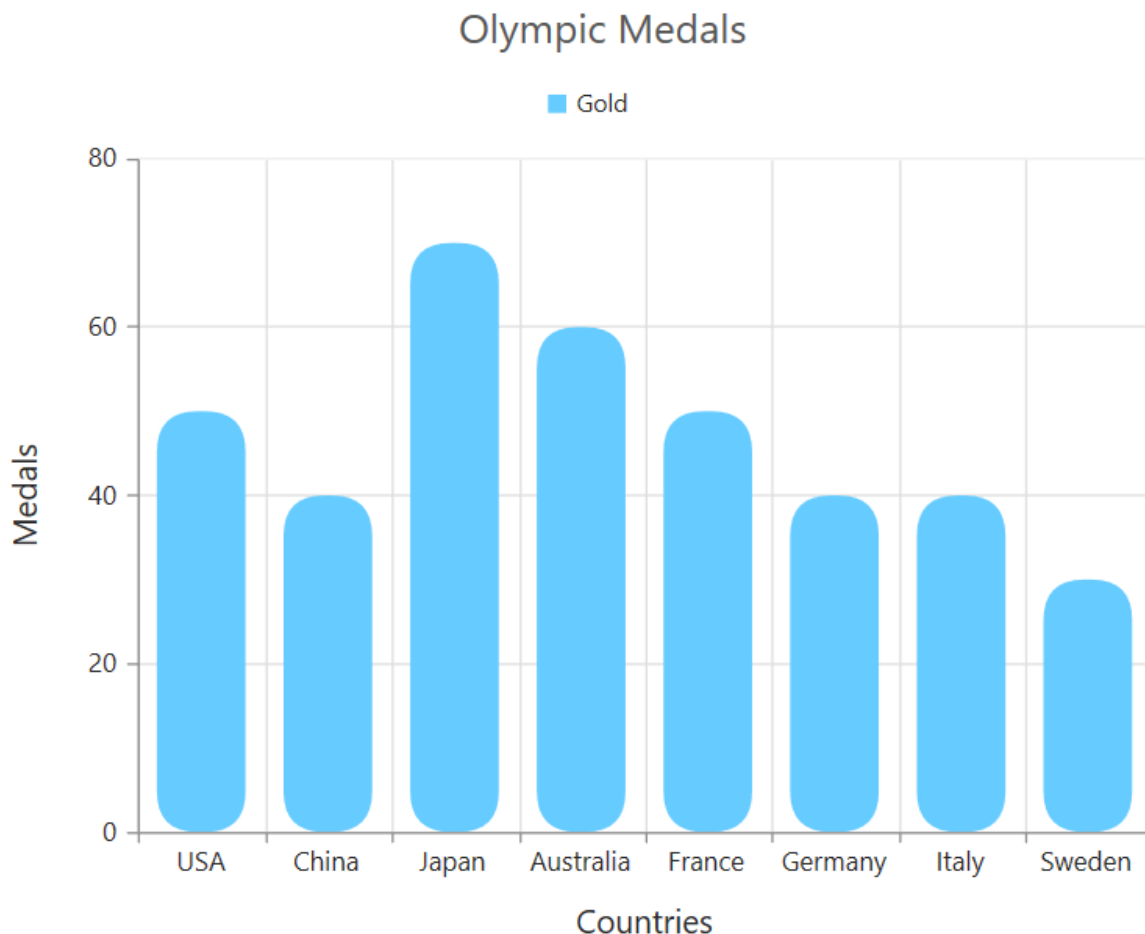


#### Column with rounded corners

Corners of the column chart can be customized by setting value to the [cornerRadius] property.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  //Common settings for all series  
  commonSeriesOptions: {  
    cornerRadius:20  
  },  
  //...  
});
```



#### Spacing between column series

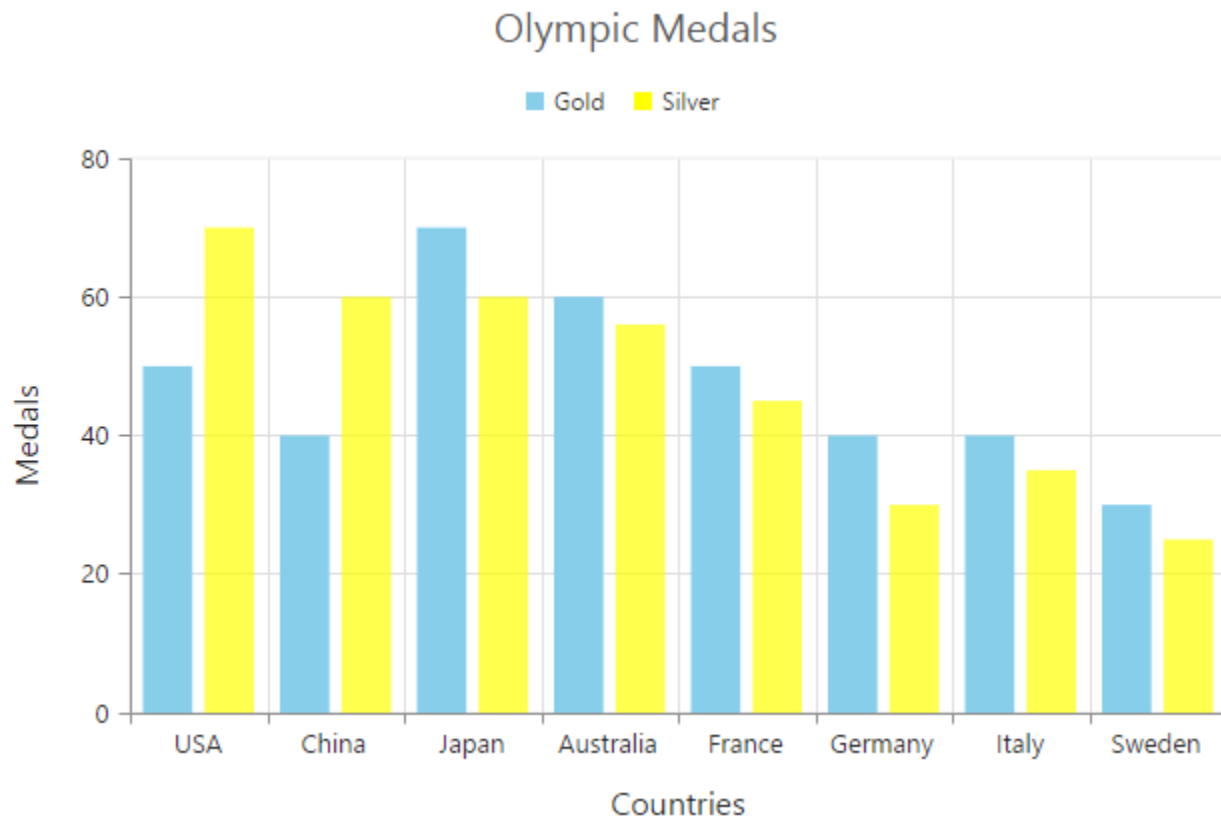
Spacing between column type series can be customized using the [columnSpacing](#) property. Default value of [columnSpacing](#) is 0. Value ranges from 0 to 1. Here 1 corresponds to 100% available space and 0 corresponds to 0% available space.

**Note:** Column spacing will also affect the width of the column. For example, setting 20% spacing and 100% width will render columns with 80% of total width. This is also applicable for StackingColumn, StackingColumn100, Bar, StackingBar, StackingBar100, RangeColumn, HiLo, HiLoOpenClose, Candle and Waterfall charts.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  //Common settings for all series
  commonSeriesOptions: {
    //Spacing between column series
    columnSpacing: 0,
    //...
  },
  //Settings specific to individual series
  series: [{
    //Spacing between column series
    columnSpacing: 0.2,
```

```
//...
}],
//...
});
```



#### Cylindrical Chart

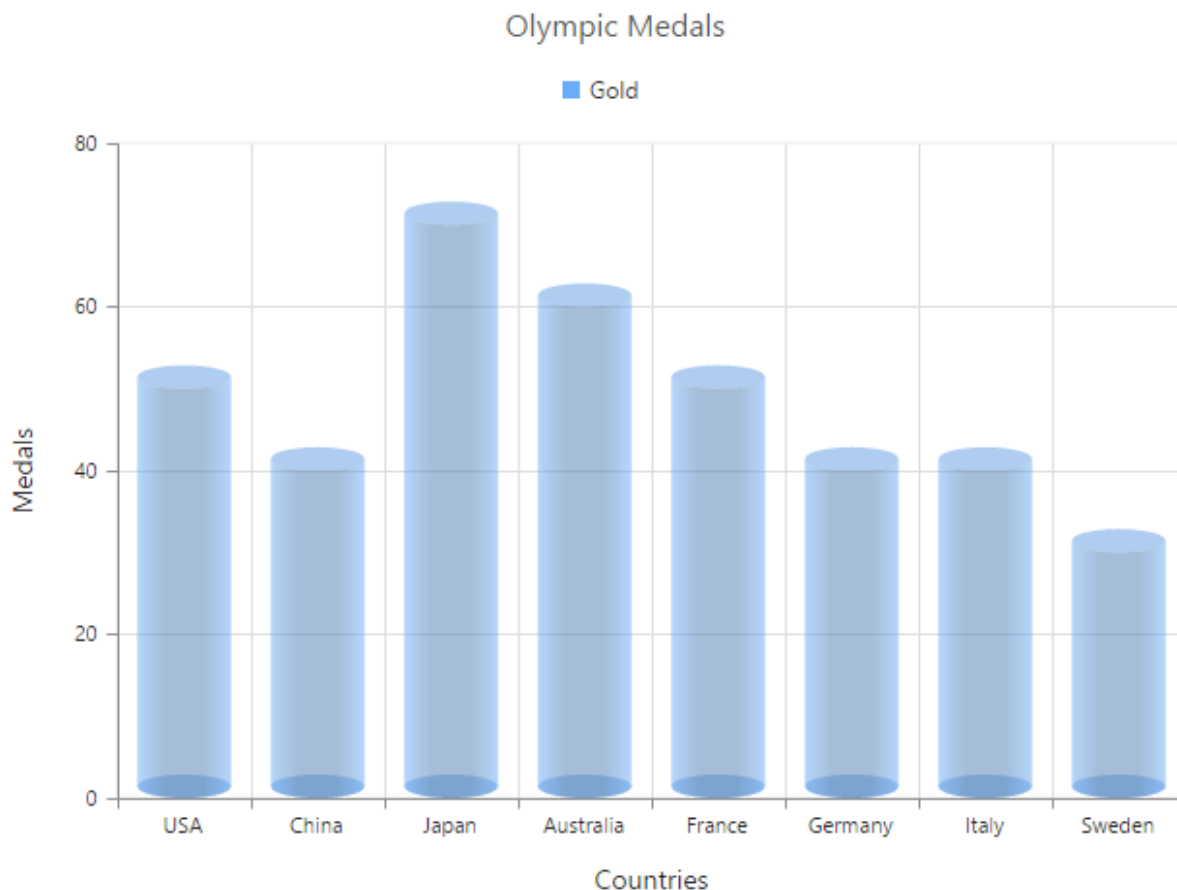
To render a cylindrical chart, set the [columnFacet](#) property as "cylinder" in the chart series along with the series type.

The following chart types can be rendered as cylinder in both 2D and in 3D view.

- Column Chart, Bar Chart, Stacked Column Chart, Stacked Bar Chart, 100% Stacked Column Chart, 100% Stacked Bar Chart.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //To change the shape of the series
    columnFacet: 'cylinder',
    type: 'column',
    // ...
  }],
  // ...
});
```



### RangeColumn Chart

To render a Range Column Chart, set the [type](#) as “**rangeColumn**” in the chart series. To change the RangeColumn color, use the [fill](#) property of the series.

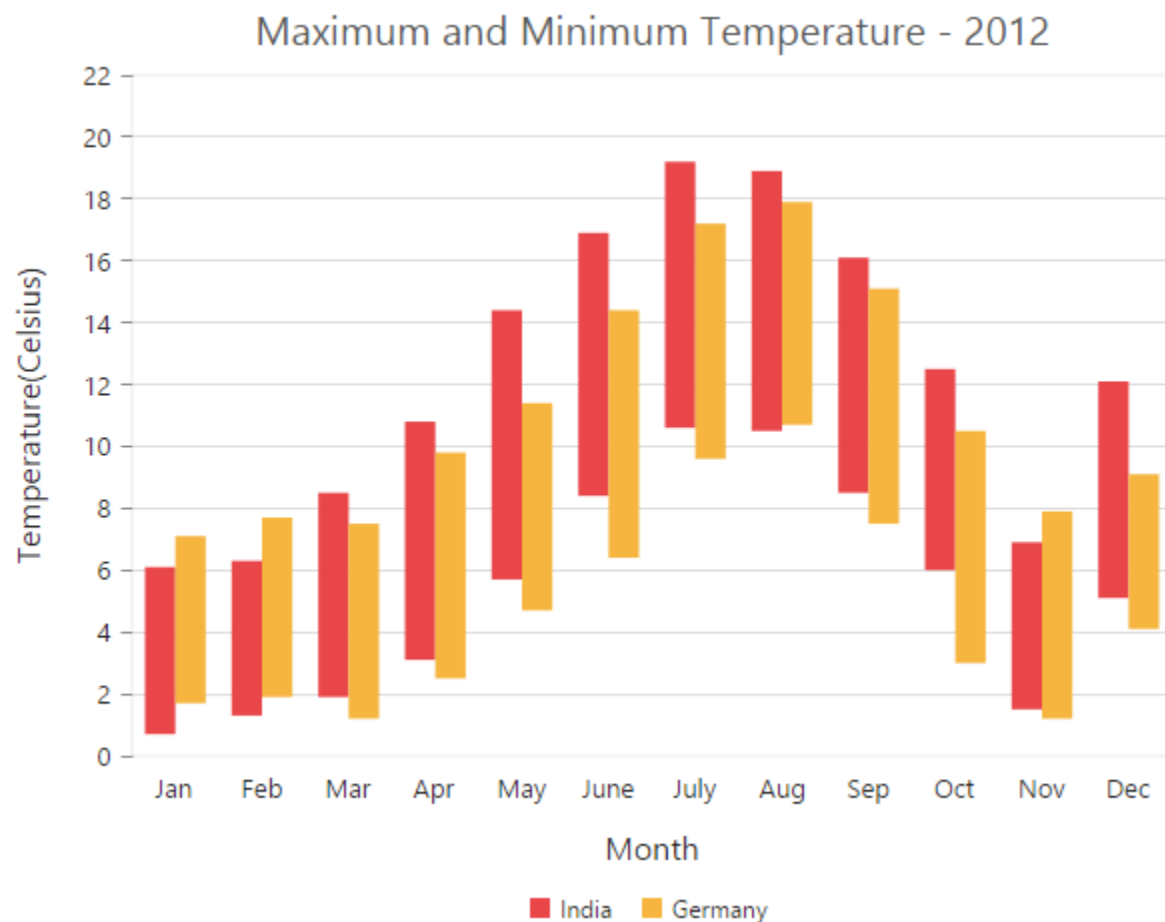
Since, the RangeColumn series requires two y values for a point, add the [high](#) and [low](#) value. High and Low value specifies the maximum and minimum range of the points.

- When you are using the [points](#) option, specify the high and low values by using the [high](#) and [low](#) option of the point.
- When you are using the [dataSource](#) option to assign the data, you have to map the fields from the dataSource that contains high and low values by using the [series.high](#) and [series.low](#) options.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Set chart type to series  
    type: 'rangeColumn',  
    fill: "#E94649",  
    //Use high and low values instead of y
```

```
points:[{ high: 6.1, low:0.7 },
// ...
],
// ...
}],
// ...
});
```

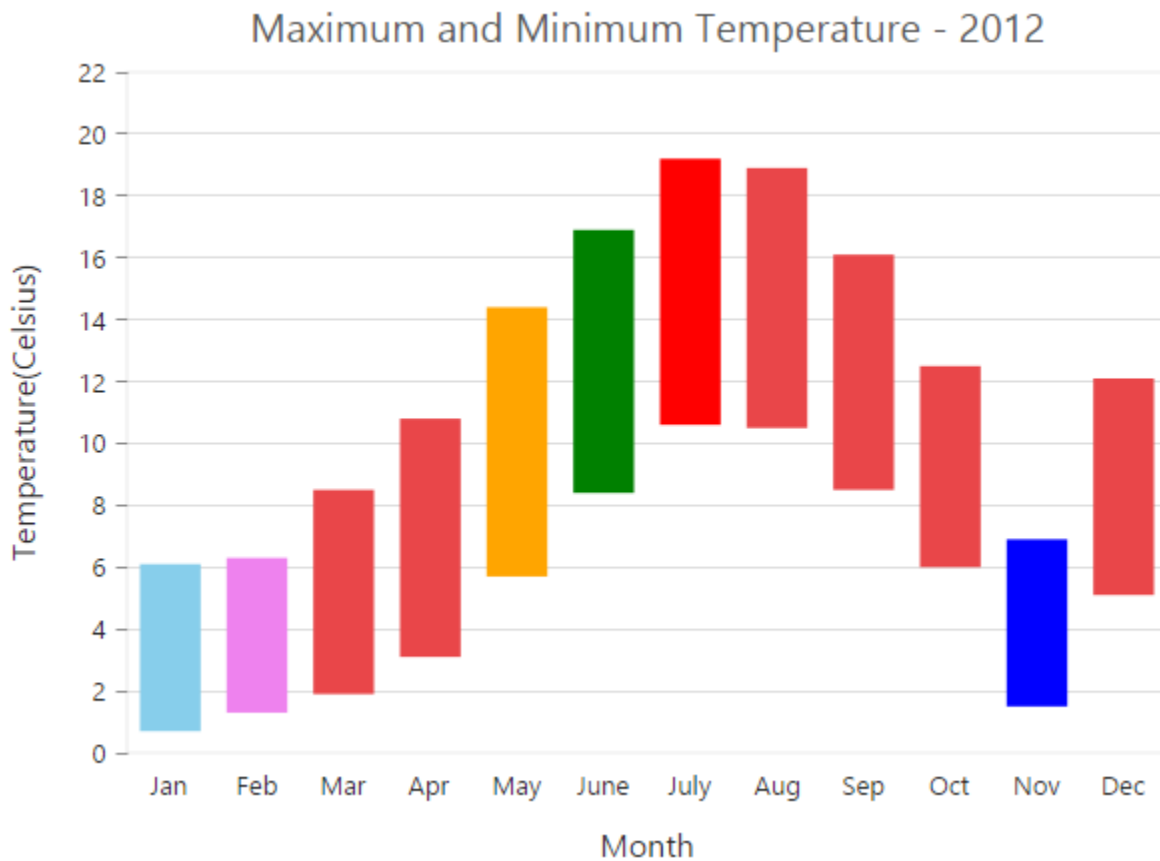


#### Change a point color

To change the color of a range column, you can use the [fill](#) property of point.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
series: [{
// Change the color of a range column
points:[{ fill: 'skyblue' },
// ...
],
// ...
}],
// ...
});
```

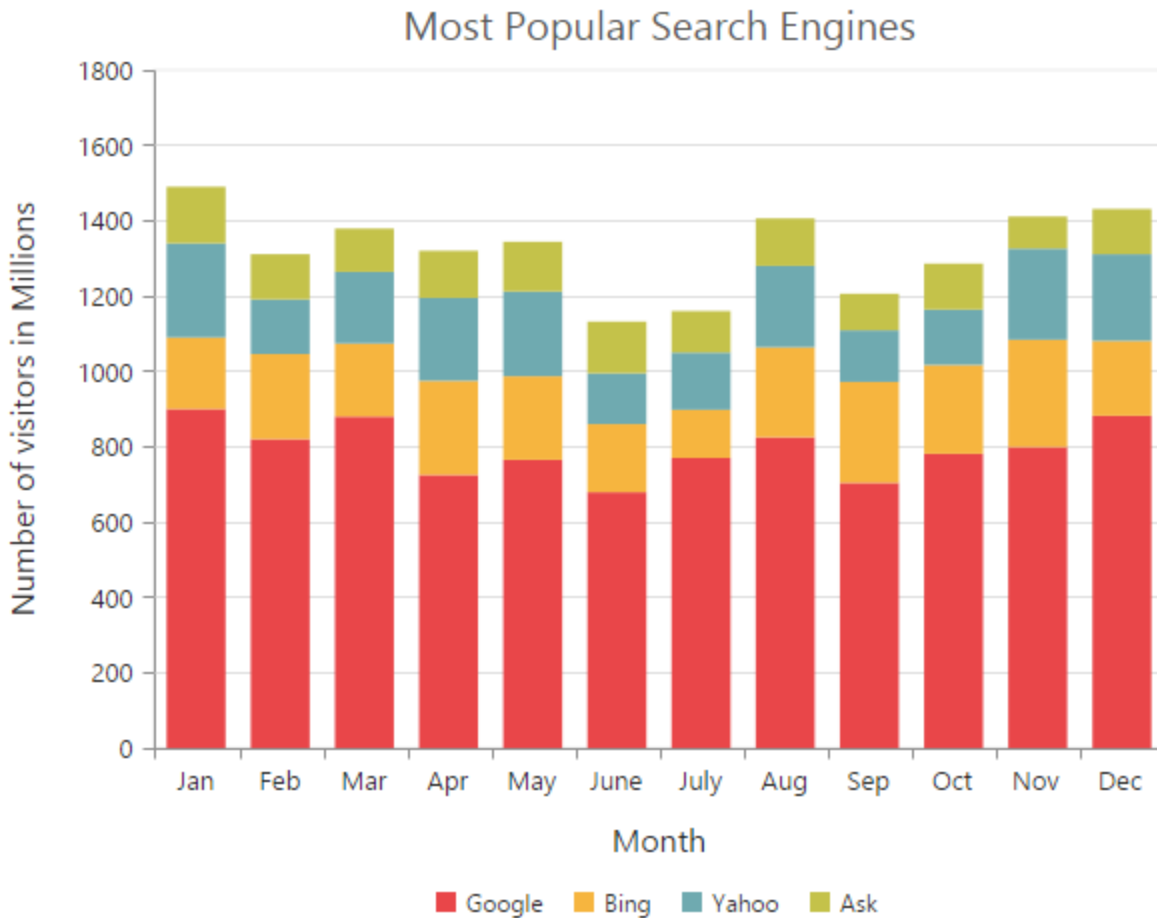


### Stacked Column Chart

To render a Stacked Column Chart, set the [type](#) as “**stackingColumn**” in the chart series. To change the StackingColumn color, you can use the [fill](#) property of the series.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change type and color of the series  
    type: 'stackingColumn',  
    fill: "#E94649",  
    // ...  
  }],  
  // ...  
});
```

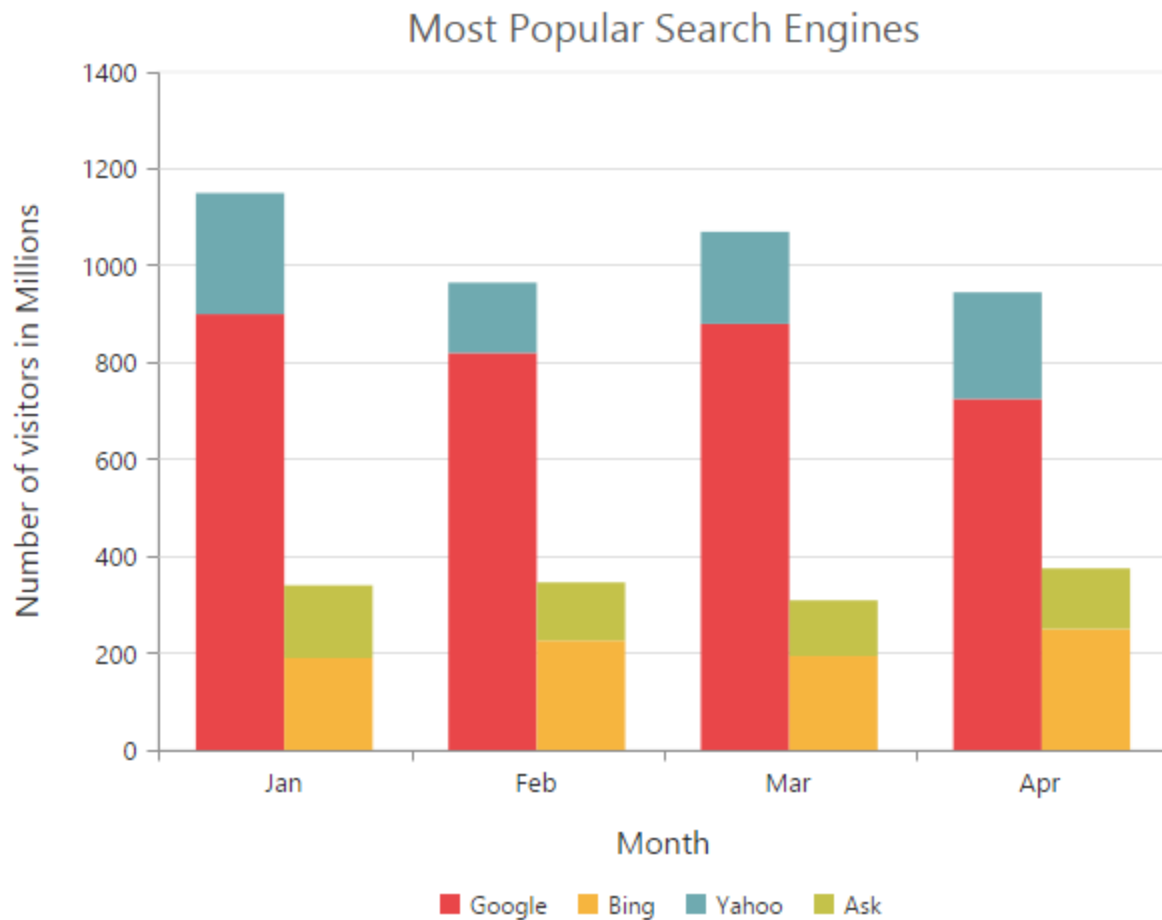


#### Cluster / Group stacked columns

You can use the [stackingGroup](#) property to group the stacked columns. Columns with same group name are stacked on top of each other.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // For grouping stacked columns
    stackingGroup: 'GroupOne'
    // ...
  },
  {
    // For grouping stacked columns
    stackingGroup: 'GroupOne'
    // ...
  }],
  // ...
});
```



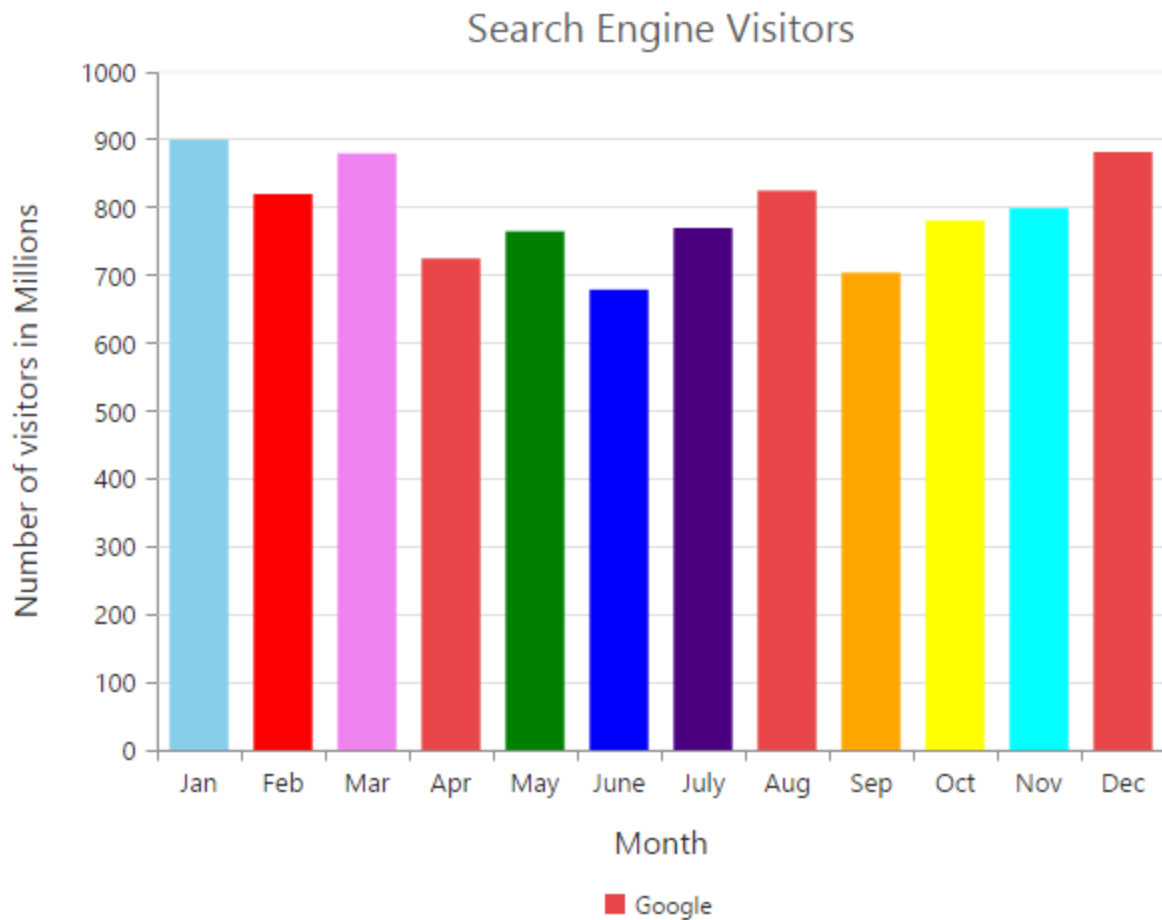
#### Change a point color

To change the color of a stacking column, you can use the [fill](#) property of the point.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    // Change the color of a stacking column  
    points: [{ fill: 'skyblue' }],  
    // ...  
  },  
  // ...  
}],  
  // ...  
});
```



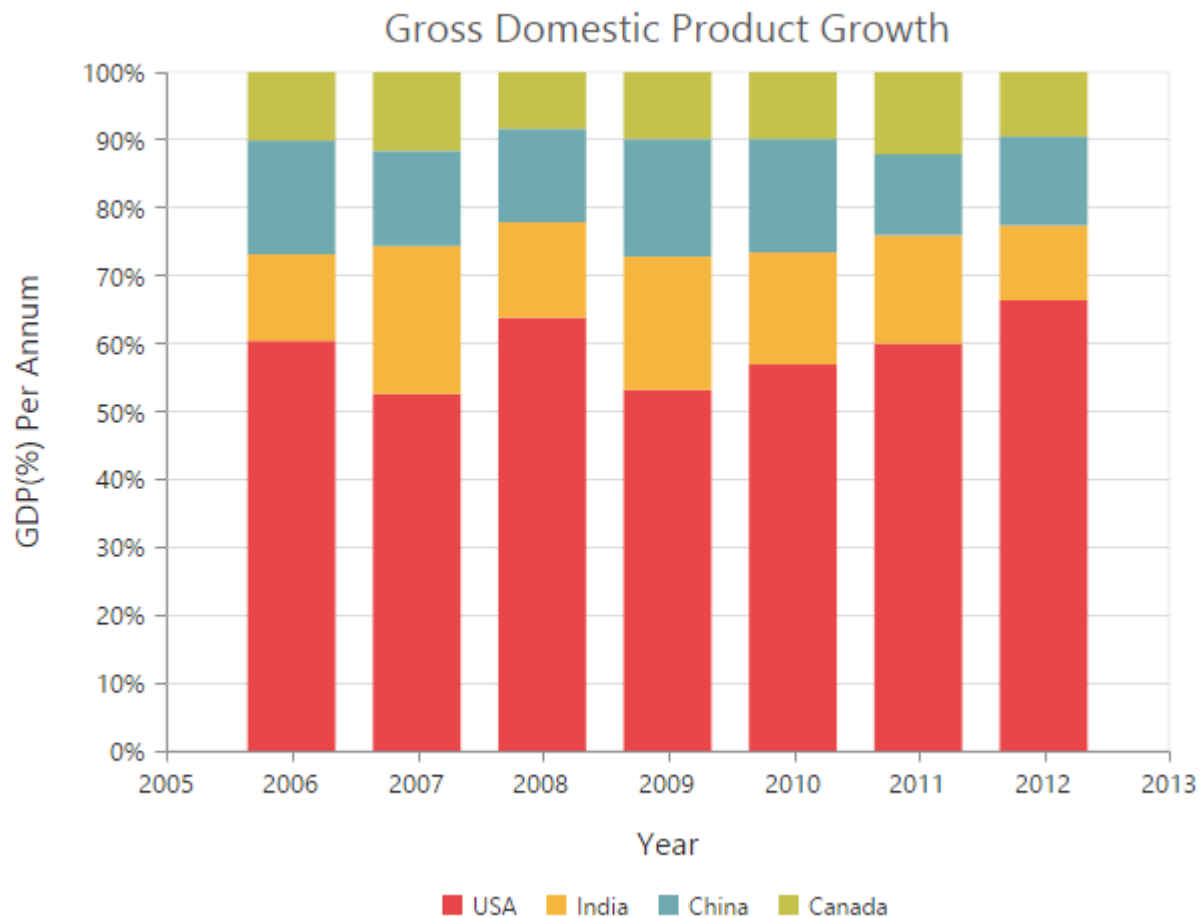


### 100% Stacked Column Chart

To render a 100% Stacked Column Chart, set the [type](#) as “**stackingColumn100**” in the chart series. To change the StackingColumn100 color, you can use the [fill](#) property of the series.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change type and color of the series  
    type: 'stackingColumn100',  
    fill: "#E94649",  
    // .....  
  }],  
  // ...  
});
```

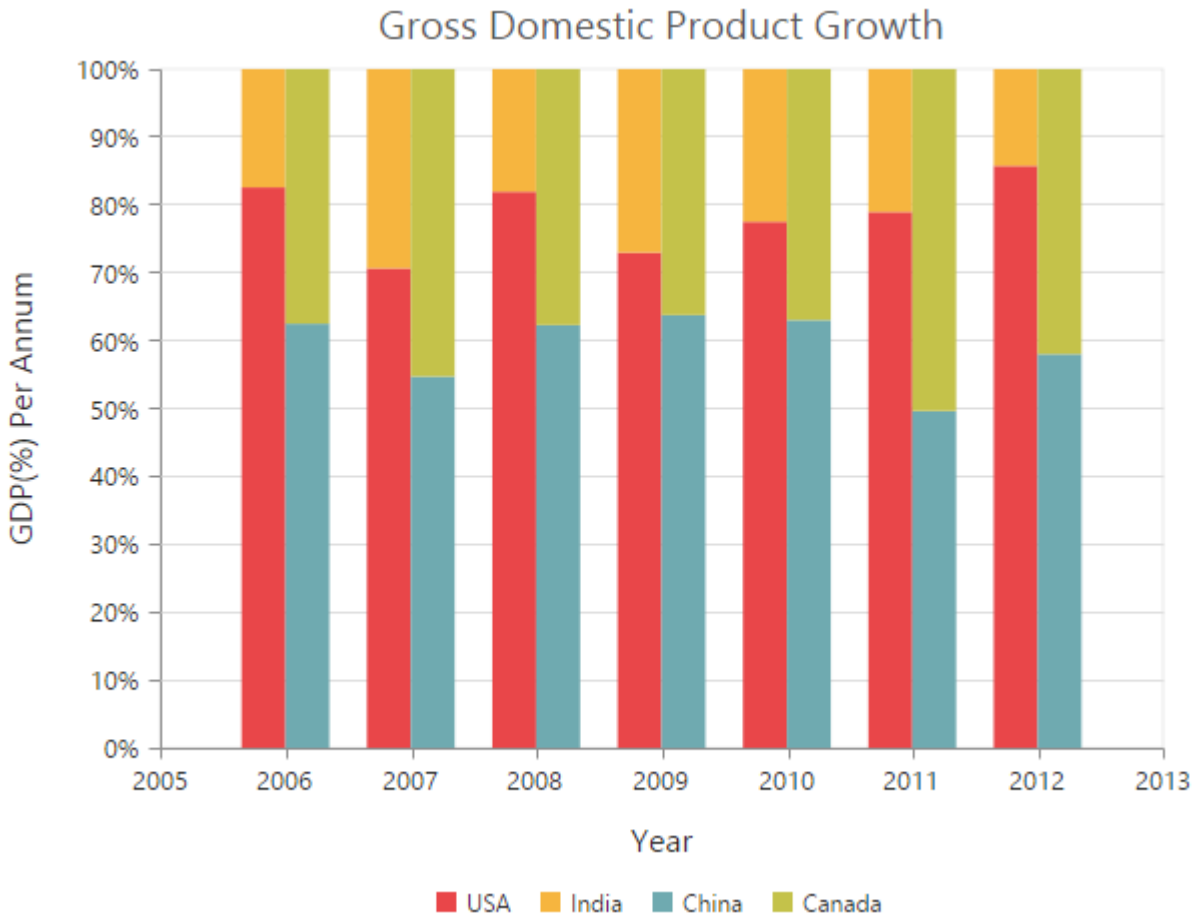


#### Cluster / Group 100% stacked columns

By using the [stackingGroup](#) property, you can group the 100% stacking columns. Columns with same group name are stacked on top of each other.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // For grouping 100% stacked columns
    stackingGroup: 'GroupOne'
    // ...
  },
  {
    // For grouping 100% stacked columns
    stackingGroup: 'GroupOne'
    // ...
  }],
  // ...
});
```

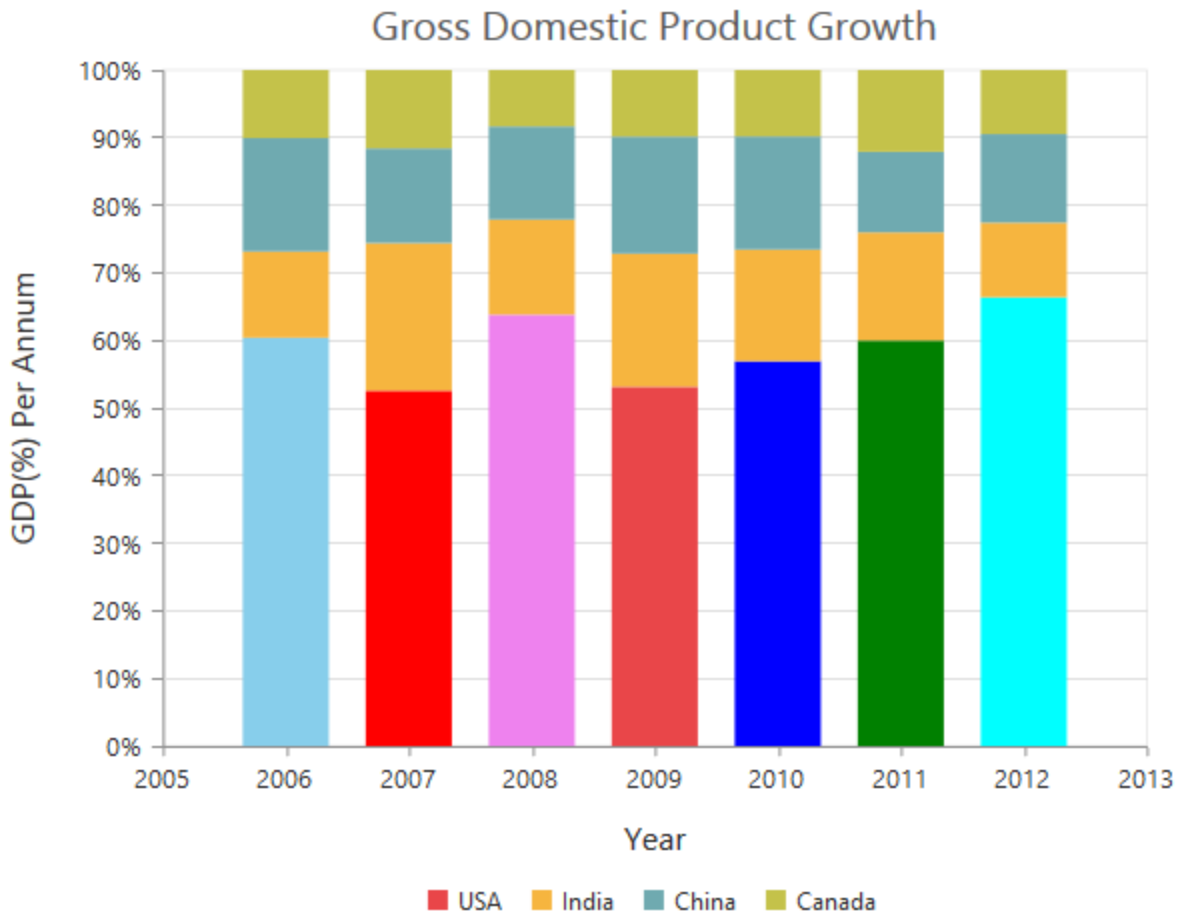


#### Change a point color

To change the color of a 100% stacking column, you can use the [fill](#) property of the point.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // Change the color of a 100% stacking column
    points: [{ fill: 'skyblue' }],
    // ...
  },
  // ...
  },
  // ...
});
```

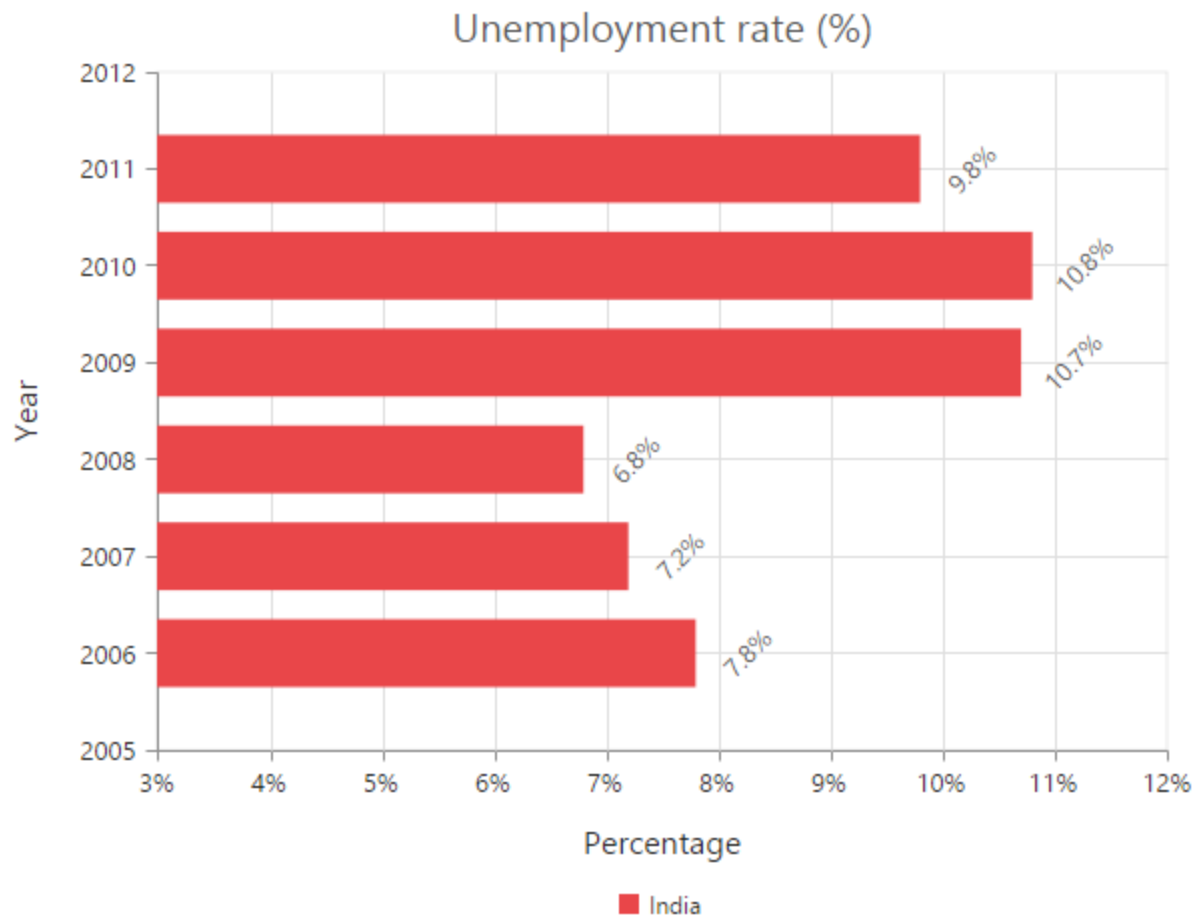


#### Bar Chart

To render a bar Chart, set the [type](#) as “**bar**” in the chart series. To change the bar color, you can use the [fill](#) property of the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Change type and color of the series
    type: 'bar',
    fill: "#E94649",
    // ...
  }],
  // ...
});
```

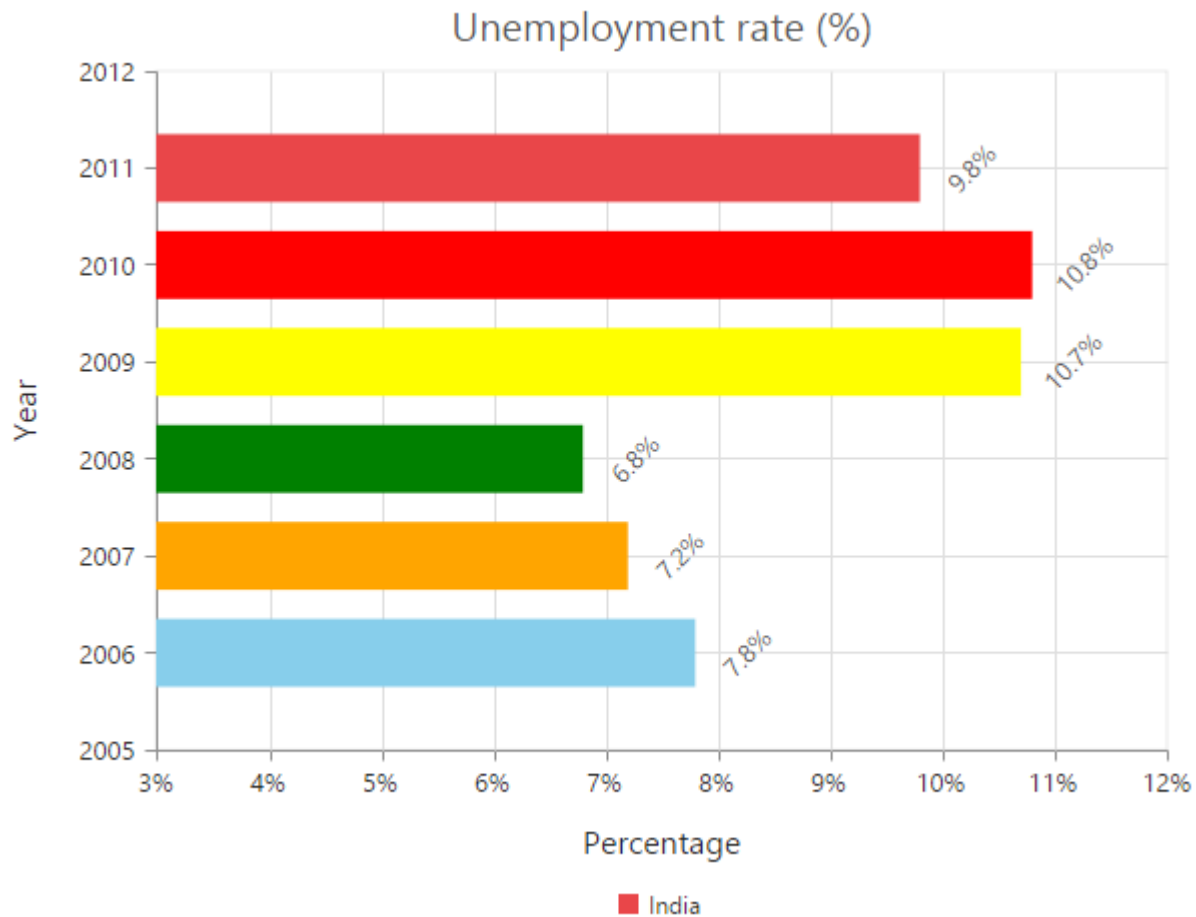


*Change the color of a bar*

By using the [fill](#) property of the point, you can change the specific point of the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    // Change the color of a bar  
    points: [{ fill: 'skyblue' },  
    // ...  
  ],  
  // ...  
}],  
  // ...  
});
```

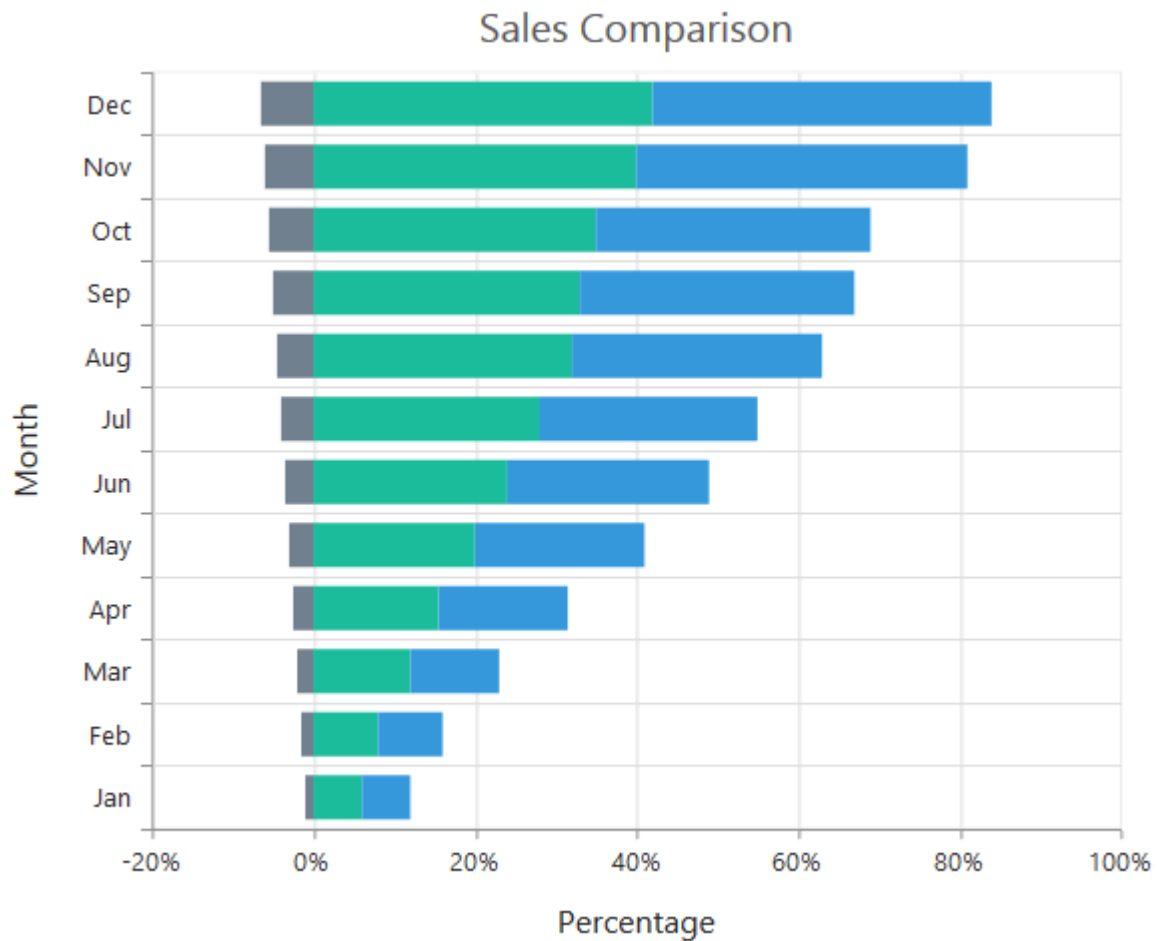


### Stacked Bar Chart

To render a Stacked Bar Chart, set the [type](#) as “**stackingBar**” in the chart series. To change the StackingBar color, you can use the [fill](#) property of the series.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Change type and color of the series.
    type: 'stackingBar',
    fill: "#E94649",
    // ...
  }],
  // ...
});
```

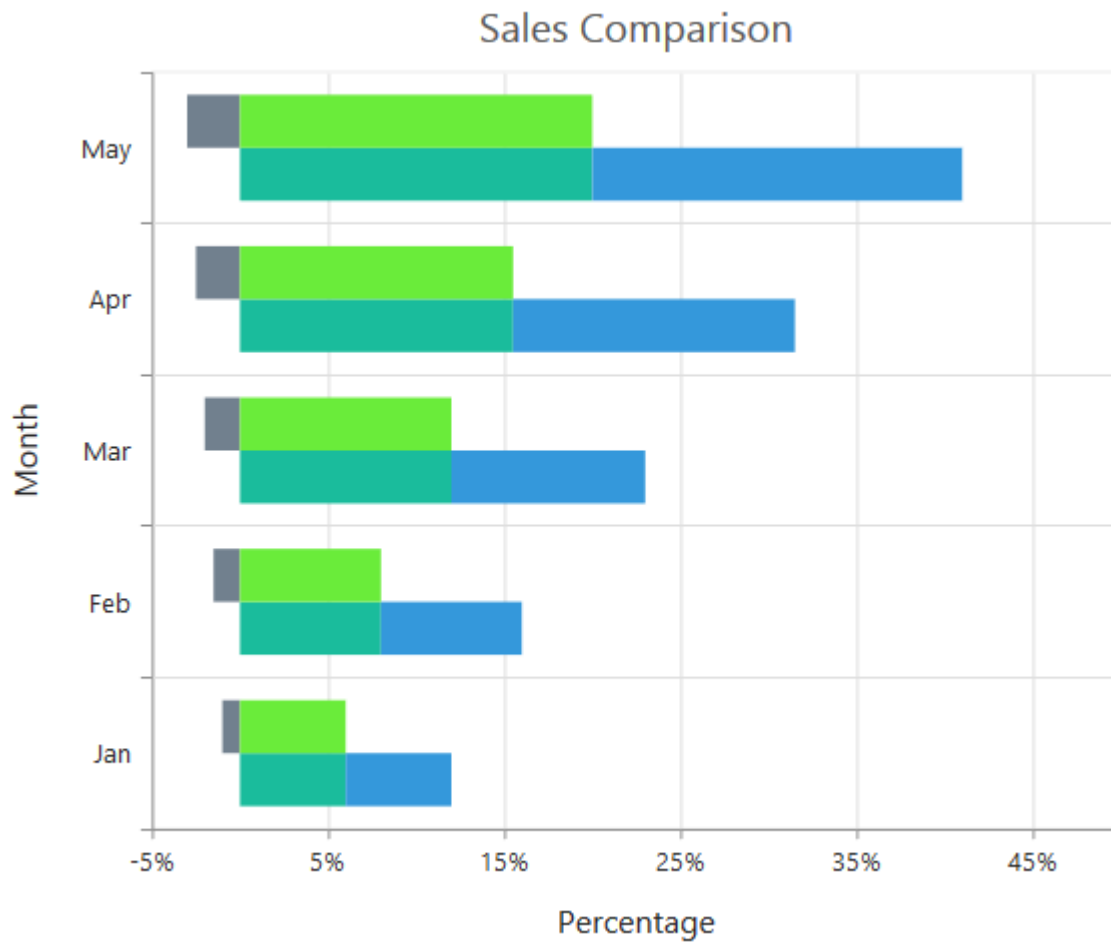


#### Cluster / Group stacked bars

You can use the [stackingGroup](#) property to group the stacking bars with the same group name.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // For grouping stacked bar
    stackingGroup: 'GroupOne'
    // ...
  },
  {
    // For grouping stacked bar
    stackingGroup: 'GroupOne'
    // ...
  }],
  // ...
});
```



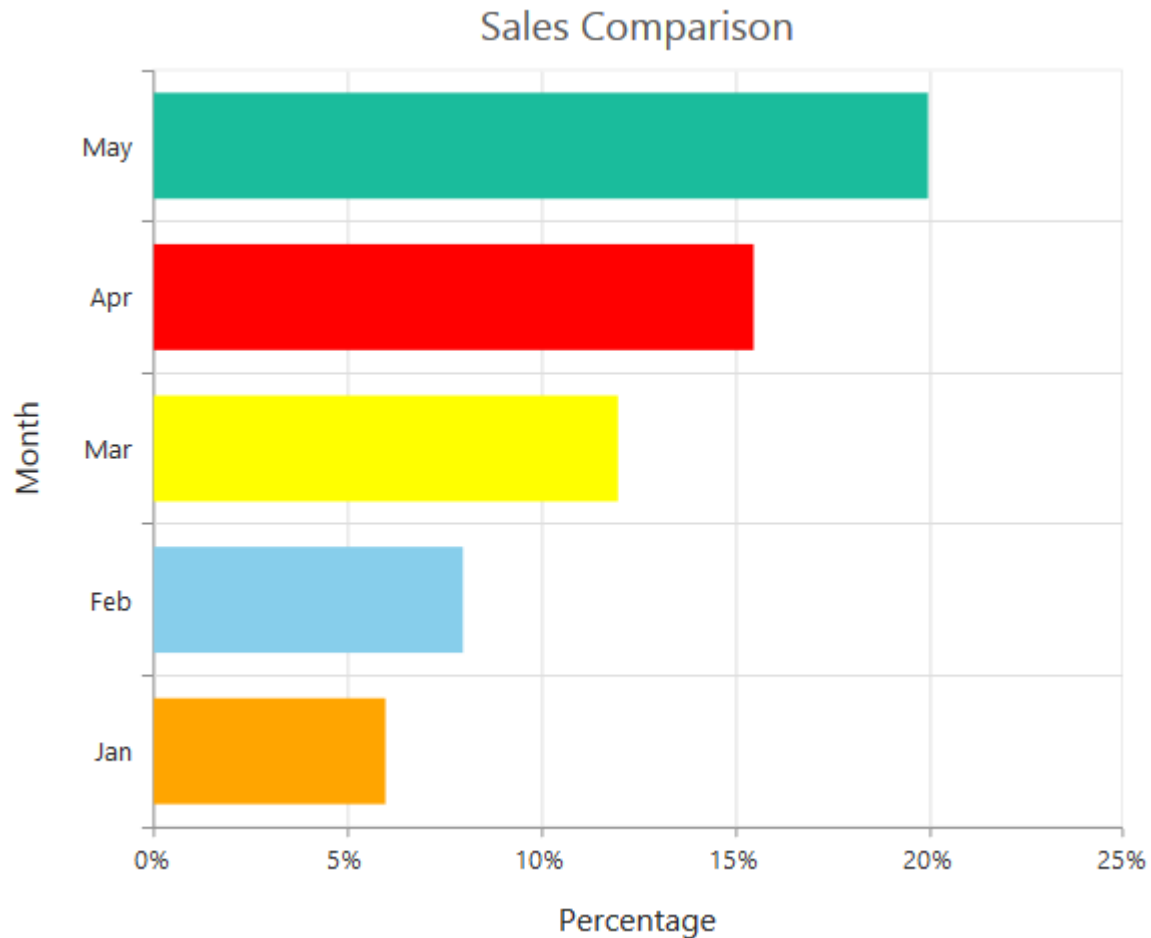
#### Change a point color

You can change the color of a stacking bar by using the [fill](#) property of the point.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // Change the color of a stacking bar
    points: [{ fill: 'skyblue' }],
    // ...
  },
  // ...
  },
  // ...
});
```



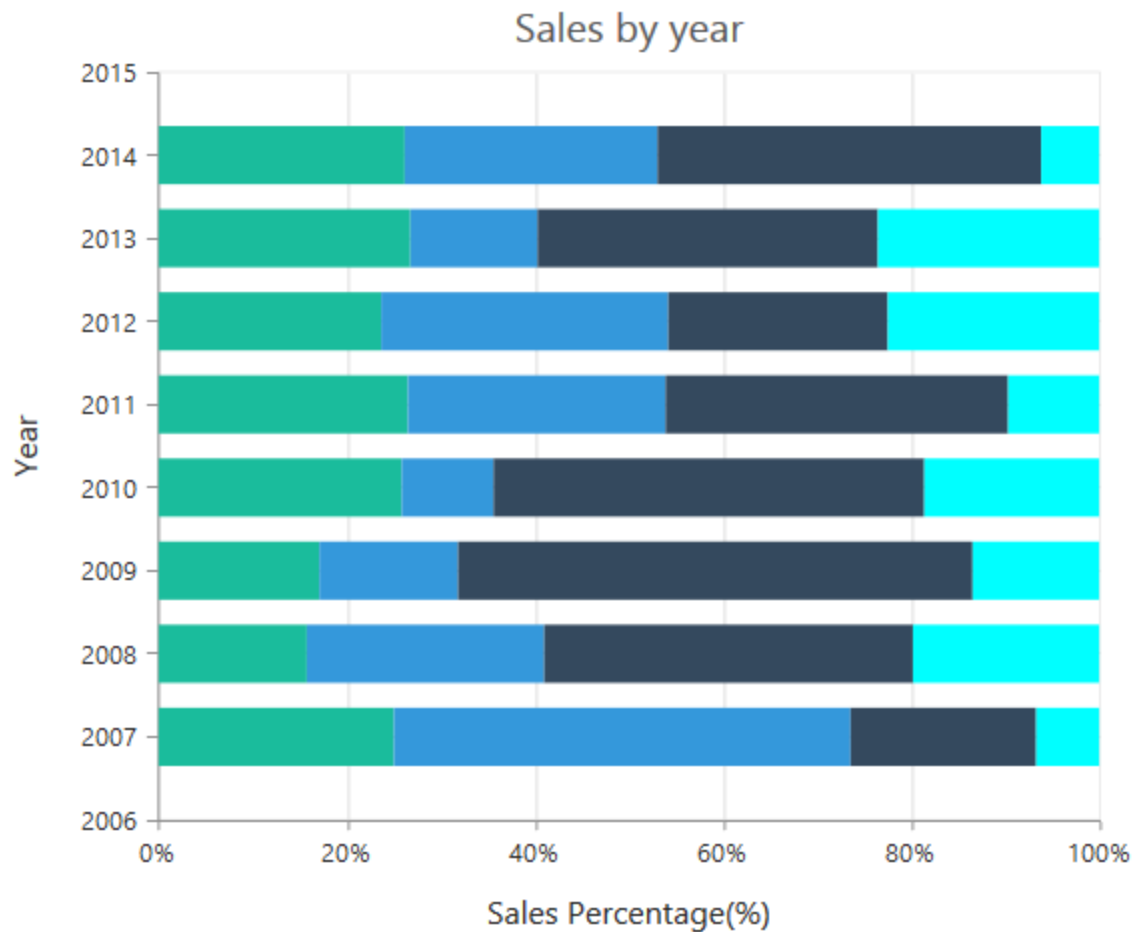


#### 100% Stacked Bar Chart

To render a 100% Stacked Bar Chart, set the [type](#) as “**stackingBar100**” in the chart series. To change the StackingBar100 color, you can use the [fill](#) property of the series.

#### JAVASCRIPT

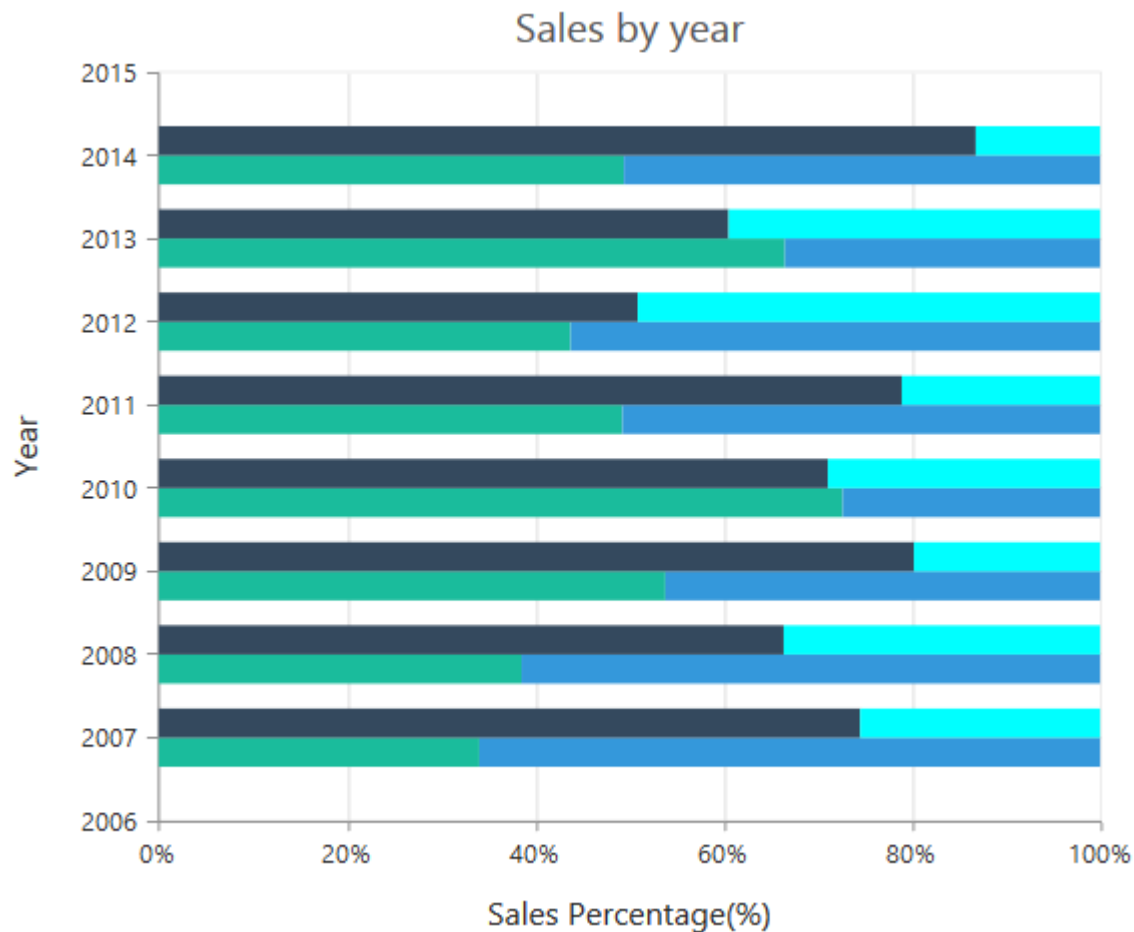
```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change type and color of the series.  
    type: 'stackingBar100',  
    fill: "#E94649",  
    // ...  
  }],  
  // ...  
});
```



By using the [stackingGroup](#) property, you can group the 100% stacking bars with the same group name.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // For grouping 100% stacked bar
    stackingGroup: 'GroupOne'
    // ...
  },
  {
    // For grouping 100% stacked bar
    stackingGroup: 'GroupOne'
    // ...
  }],
  // ...
});
```

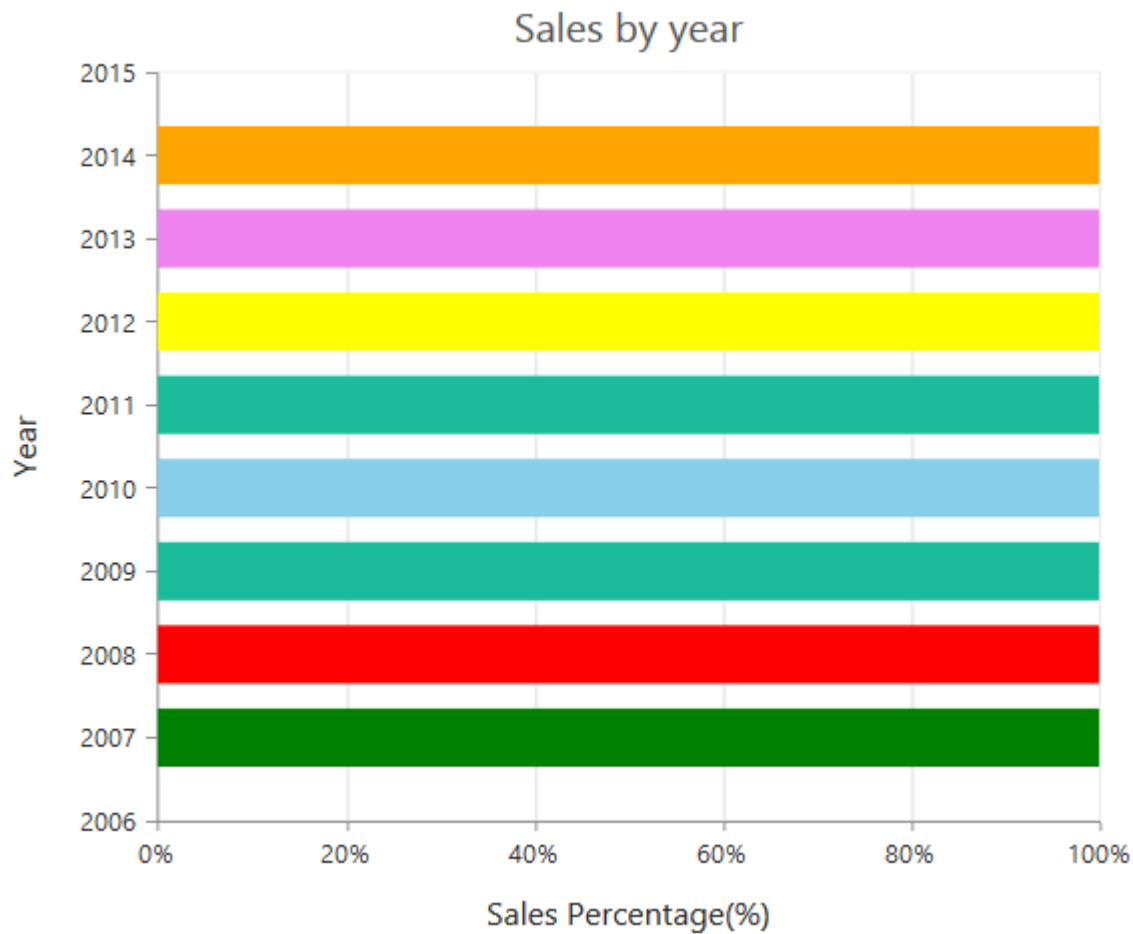


### Change a point color

To change the color of a 100% stacking bar, you can use the [fill](#) property of the point.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // Change the color of a 100% stacking bar
    points: [{ fill: 'skyblue' }],
    // ...
  },
  // ...
  },
  // ...
});
```

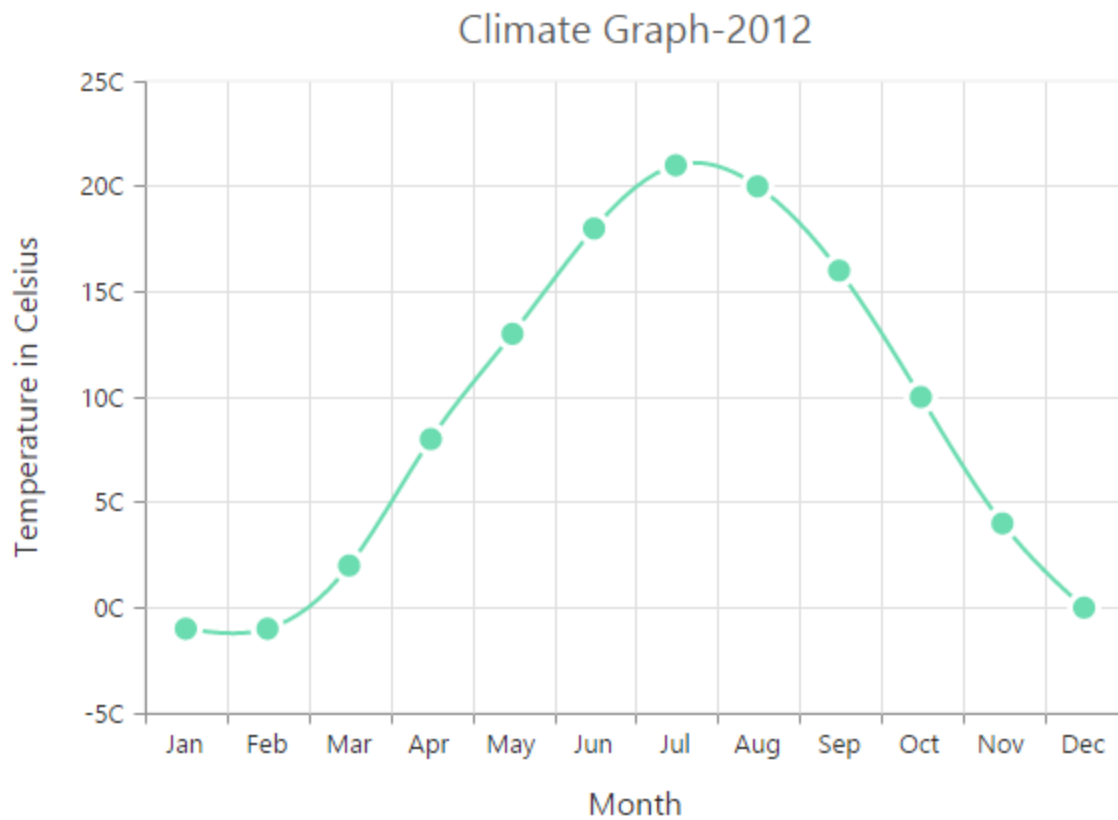


### Spline Chart

To render a Spline Chart, set the [type](#) as “**spline**” in the chart series. To change the Spline segment color, you can use the [fill](#) property of the series.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change type and color of the series.  
    type: 'spline',  
    fill: "#6ADCB0",  
    // ...  
  }],  
  // ...  
});
```

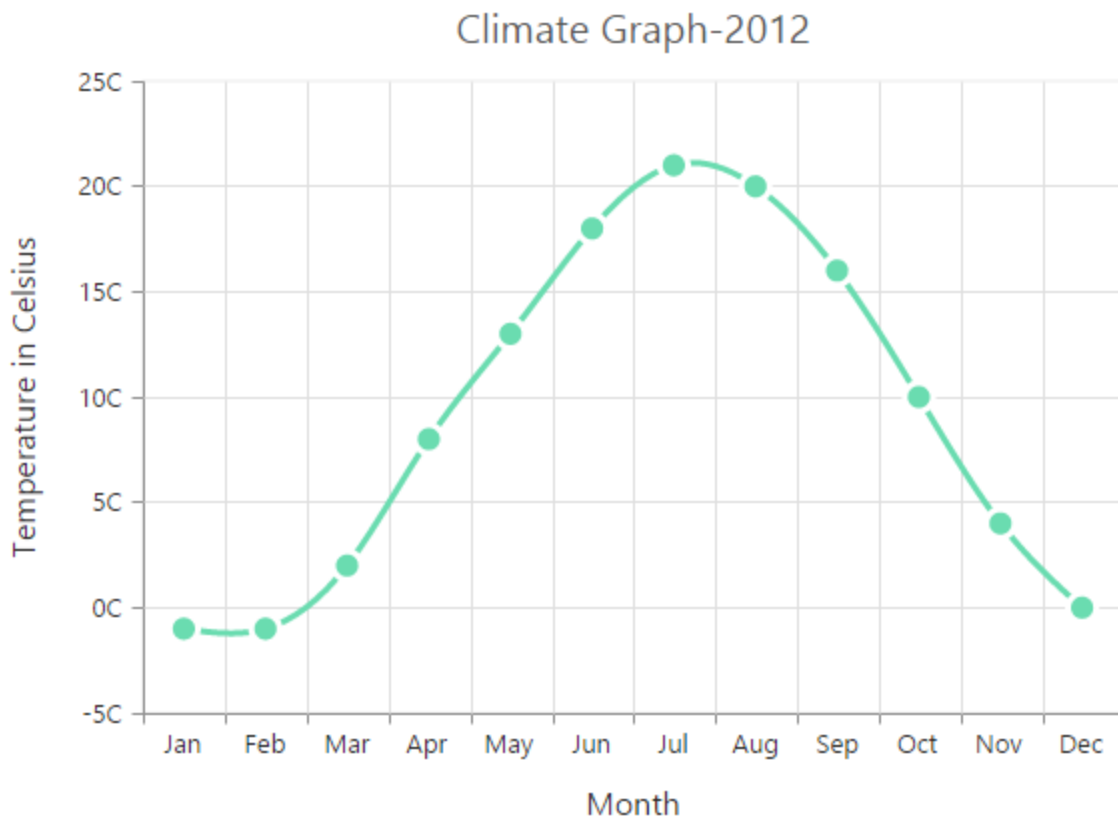


#### Change the spline width

To change the spline segment width, you can use the [width](#) property of the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change the width of spline series  
    width: 3,  
    // ...  
  }],  
  // ...  
});
```

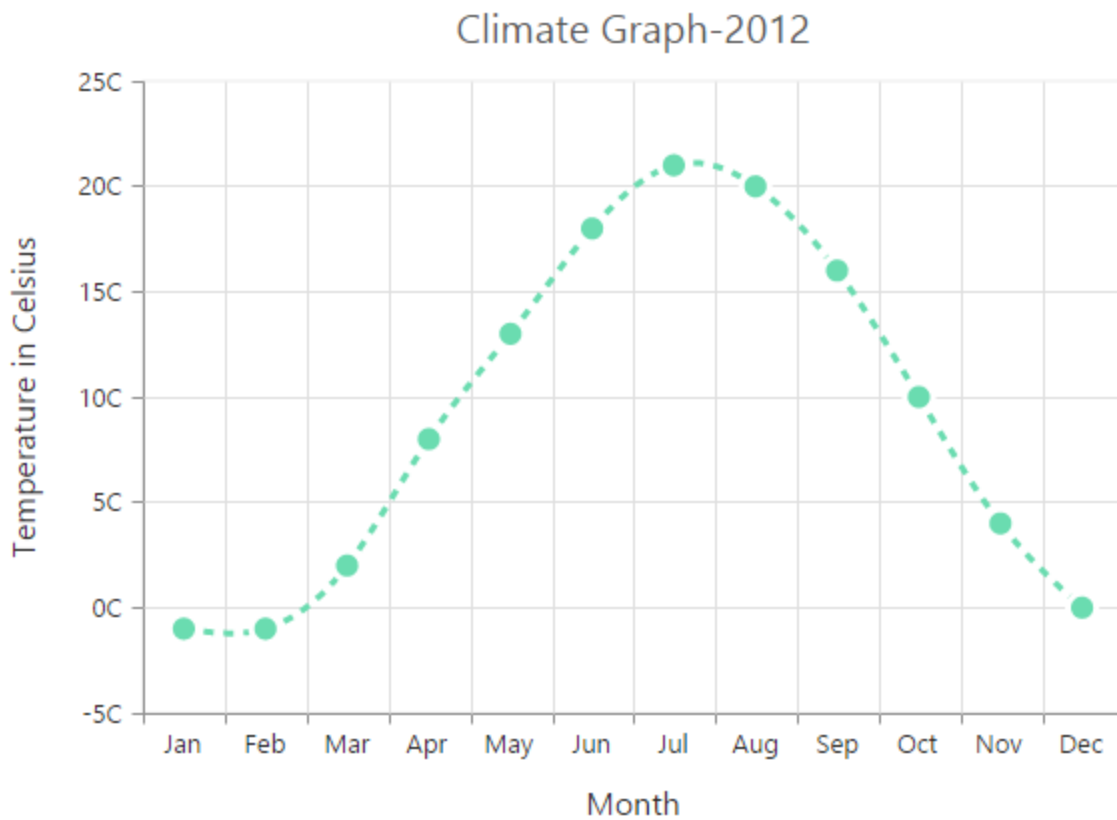


#### Dashed lines

To render the spline series with dotted lines, you can use the [dashArray](#) option of the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change dash array to display dotted or dashed splines  
    dashArray: '5,5',  
    // ...  
  }],  
  // ...  
});
```

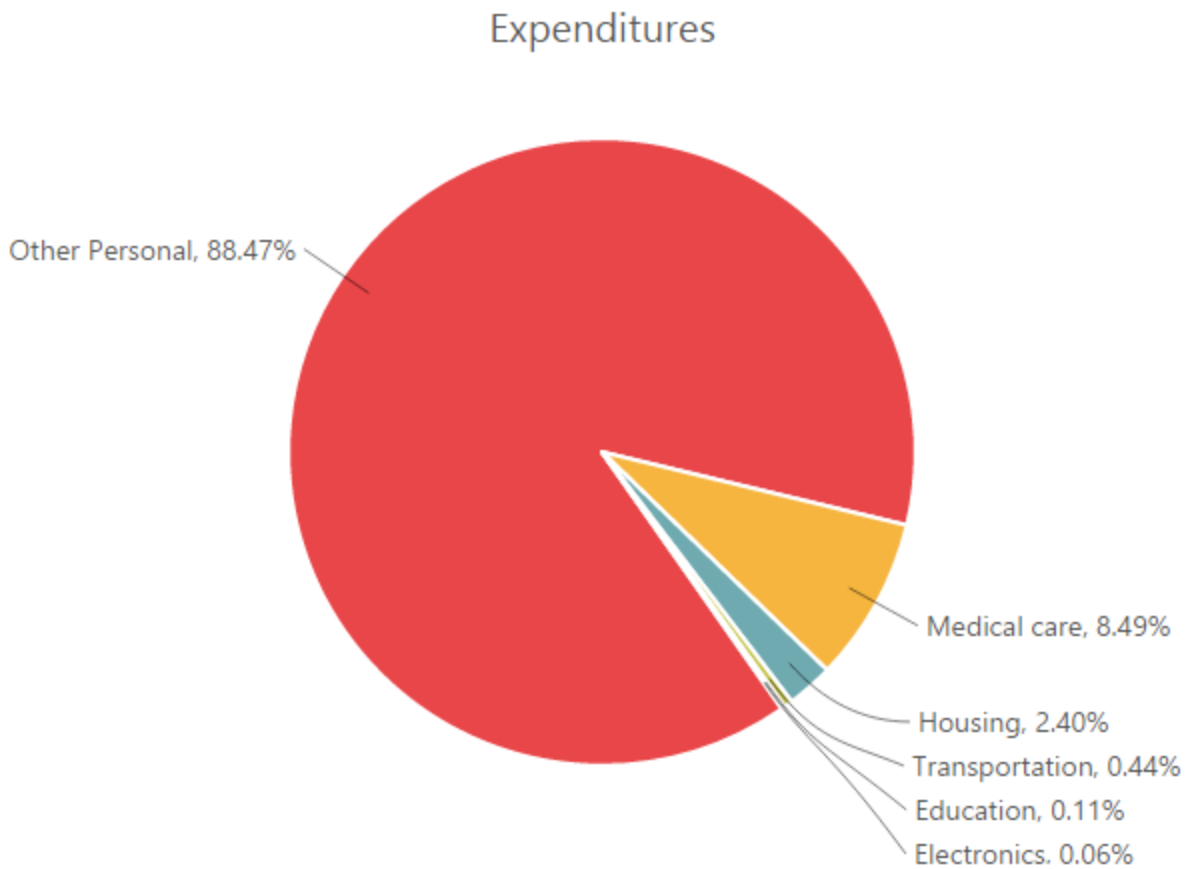


#### Pie Chart

You can create a pie chart by setting the series [type](#) as “**pie**” in the chart series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Set chart type to series  
    type: 'pie',  
    // ...  
  }],  
  // ...  
});
```



#### *Change the pie size*

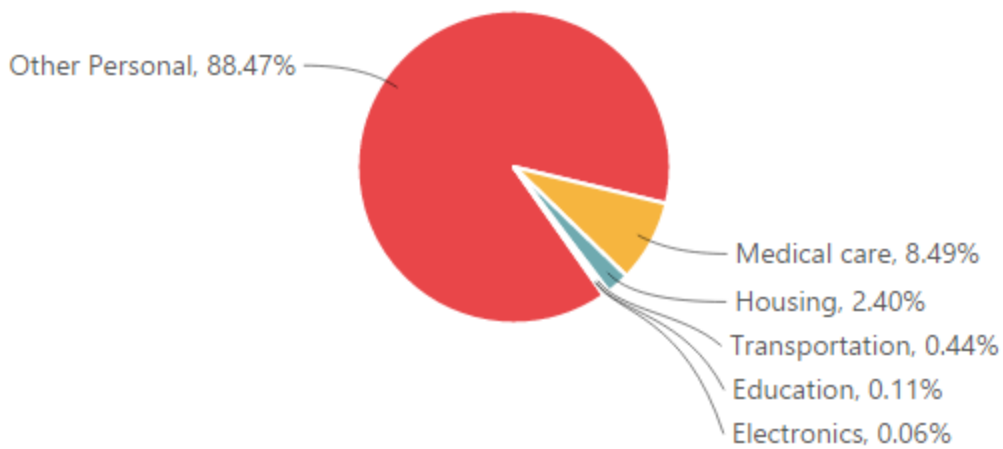
You can use the [pieCoefficient](#) property to change the diameter of the Pie chart with respect to the plot area. It ranges from 0 to 1 and the default value is **0.8**.

#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change pie chart coefficient value  
    pieCoefficient: 0.4,  
    // ...  
  }],  
  // ...  
});
```



## Expenditures

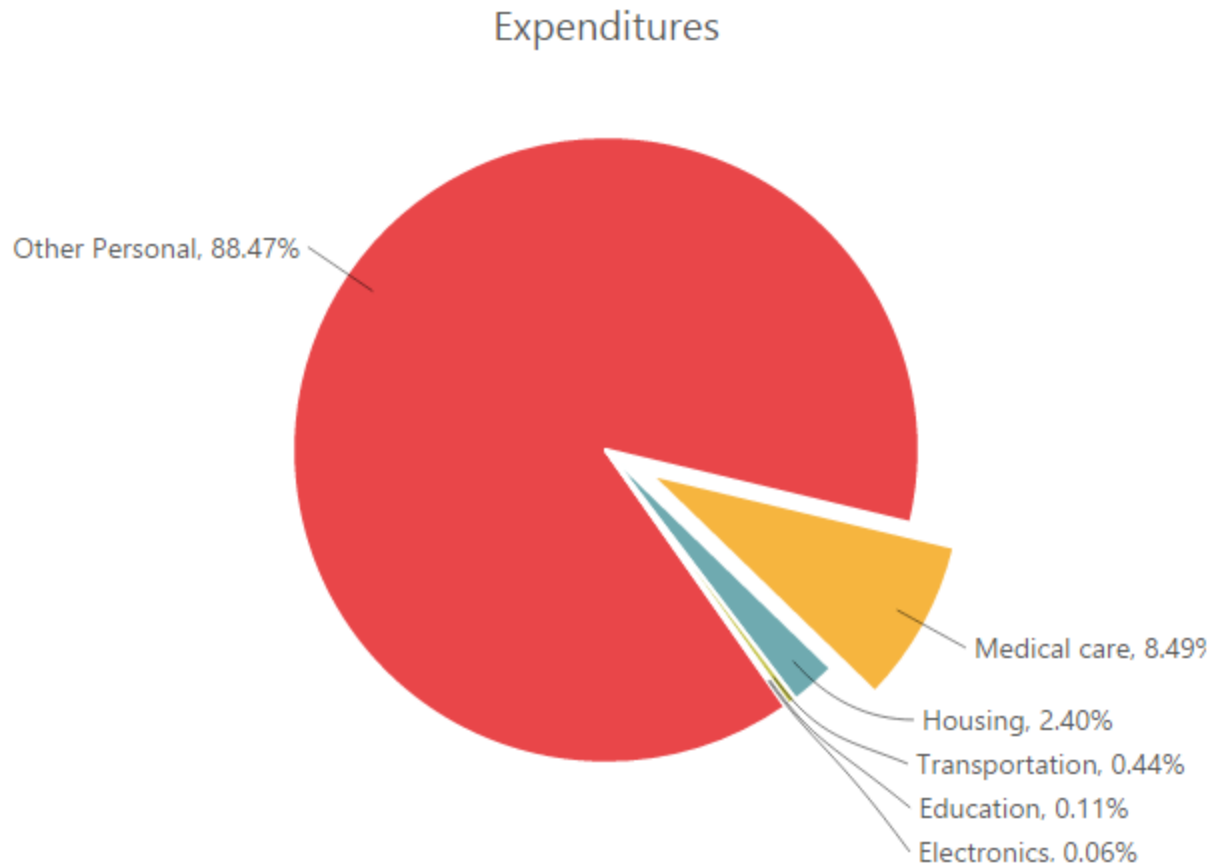


### *Explode a pie segment*

You can explode a pie segment on the chart load by using the [explodeIndex](#) of the series.

### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Set point index value to explode the pie segment.  
    explodeIndex: 1,  
    // ...  
  }],  
  // ...  
});
```

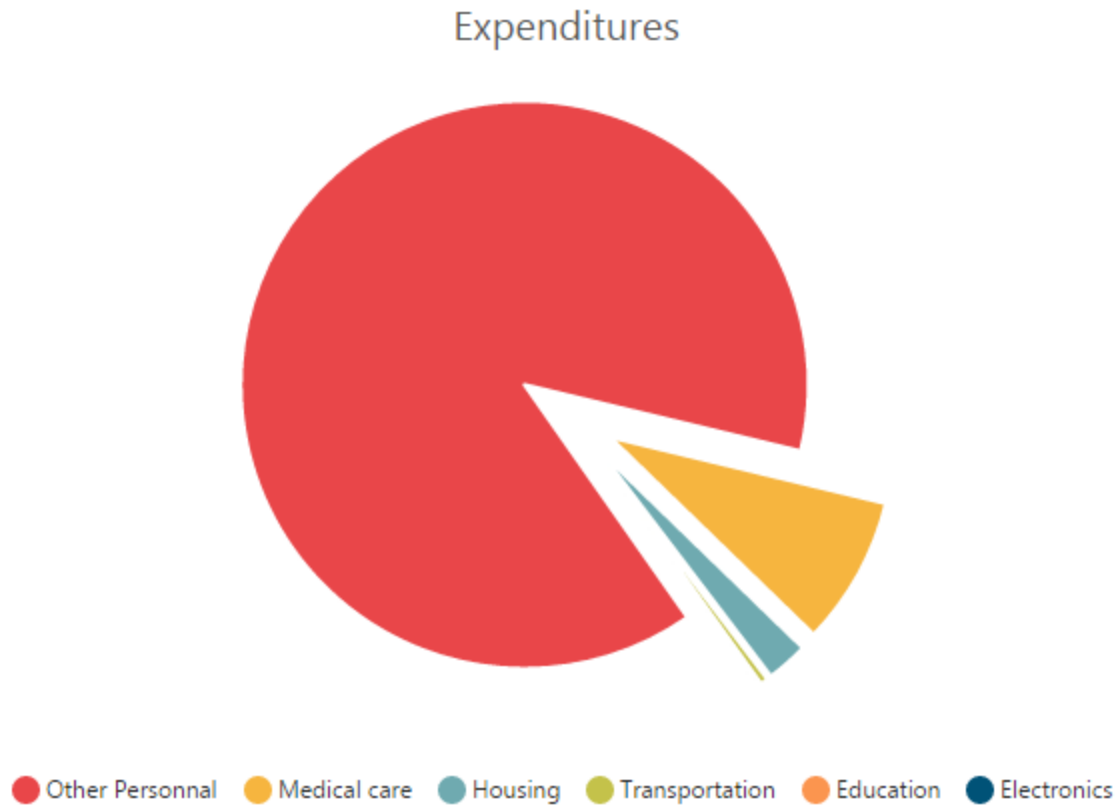


#### *Explode all the segments*

To explode all the segments of the Pie chart, you can enable the [explodeAll](#) property.

#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Enable explodeAll property for pie chart.  
    explodeAll: true,  
    // ...  
  }],  
  // ...  
});
```

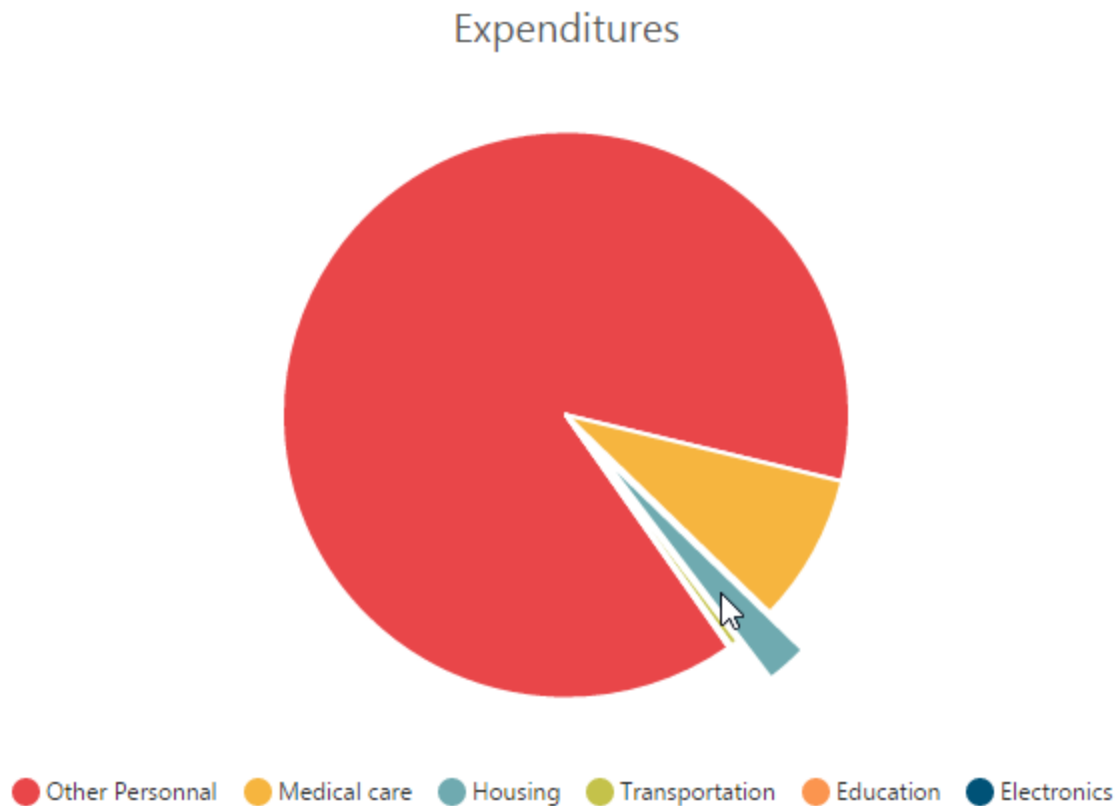


*Explode a pie segment on mouse over*

To explode a pie segment on a mouse over, you can enable the [explode](#) property of the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Enable pie explode option on mouse over the chart  
    explode: true,  
    // ...  
  }],  
  // ...  
});
```



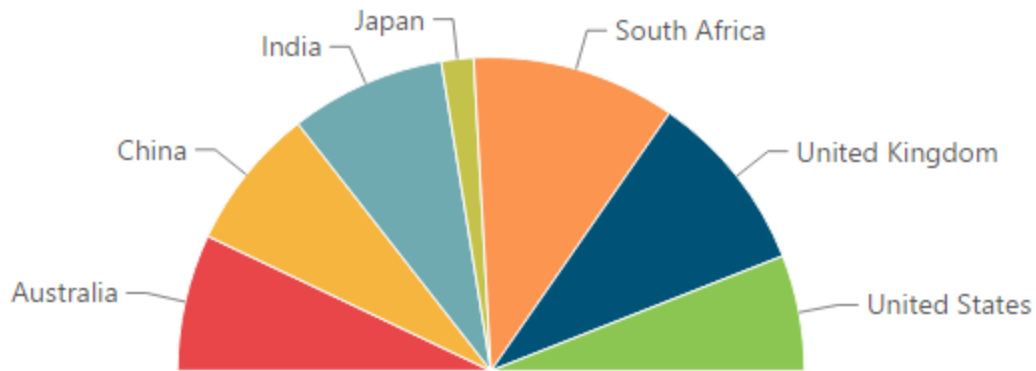
#### Sector of Pie

EjChart allows you to render all the data points/segments in the semi-pie, quarter-pie or in any sector by using the [startAngle](#) and [endAngle](#) options.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    type: 'pie',  
    //Set startAngle and endAngle to draw the semi pie chart  
    startAngle: -90, endAngle: 90  
    // ...  
  }],  
  // ...  
});
```

## Agricultural land in 2011 (% of land area)

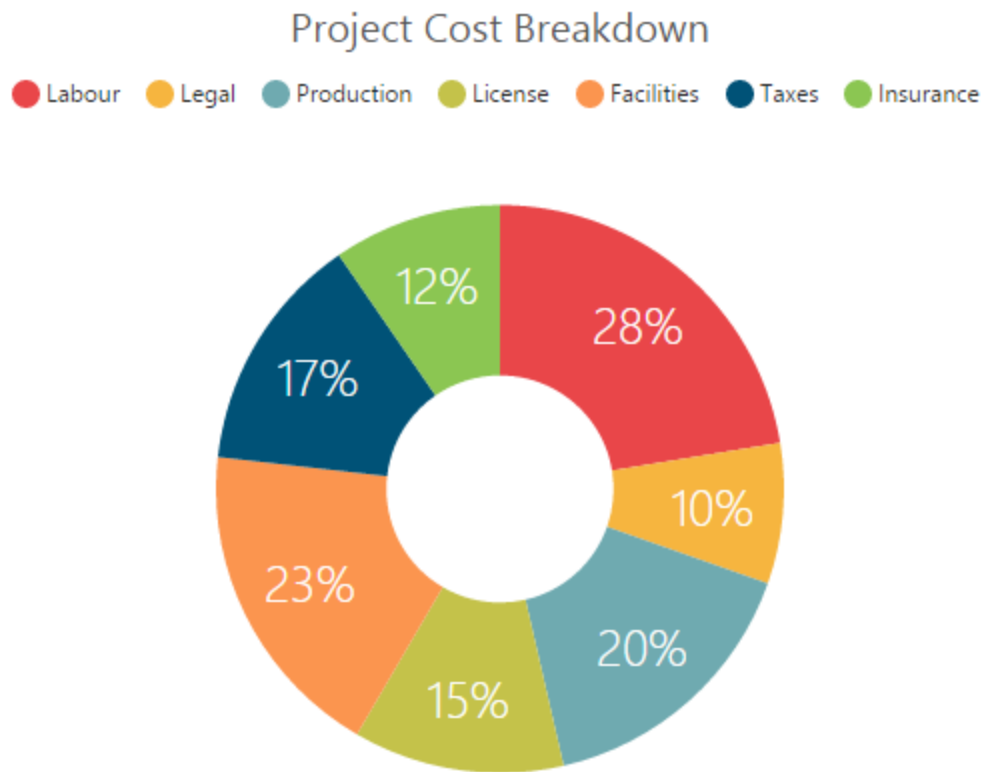


### Doughnut Chart

To create a Doughnut chart, you can specify the series [type](#) as “doughnut” in the chart series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Set chart type to series  
    type: 'doughnut',  
    // ...  
  }],  
  // ...  
});
```

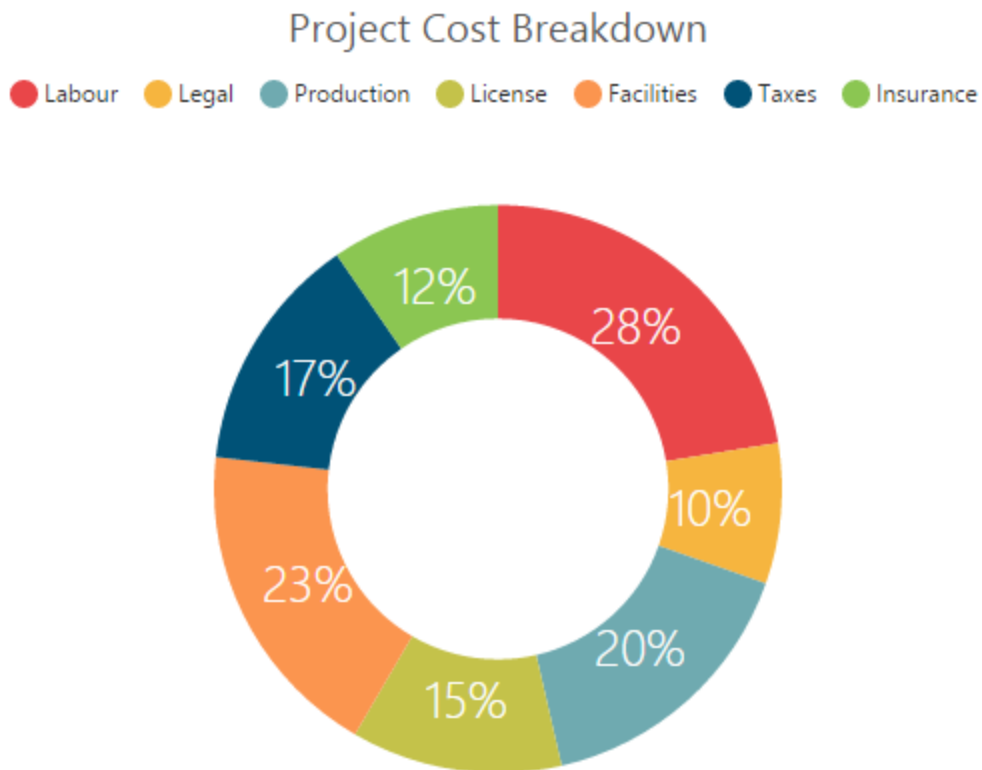


#### *Change Doughnut inner radius*

You can change the doughnut chart inner radius by using the [doughnutCoefficient](#) with respect to the plot area. It ranges from 0 to 1 and the default value is **0.4**.

#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change doughnut chart coefficient value  
    doughnutCoefficient: 0.6  
    // ...  
  }],  
  // ...  
});
```



#### *Change the doughnut size*

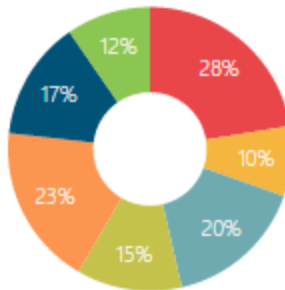
You can use the [doughnutSize](#) property to change the diameter of the Doughnut chart with respect to the plot area. It ranges from 0 to 1 and the default value is **0.8**.

#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change doughnut chart coefficient value  
    doughnutSize: 0.4,  
    // ...  
  }],  
  // ...  
});
```

## Project Cost Breakdown

● Labour ● Legal ● Production ● License ● Facilities ● Taxes ● Insurance



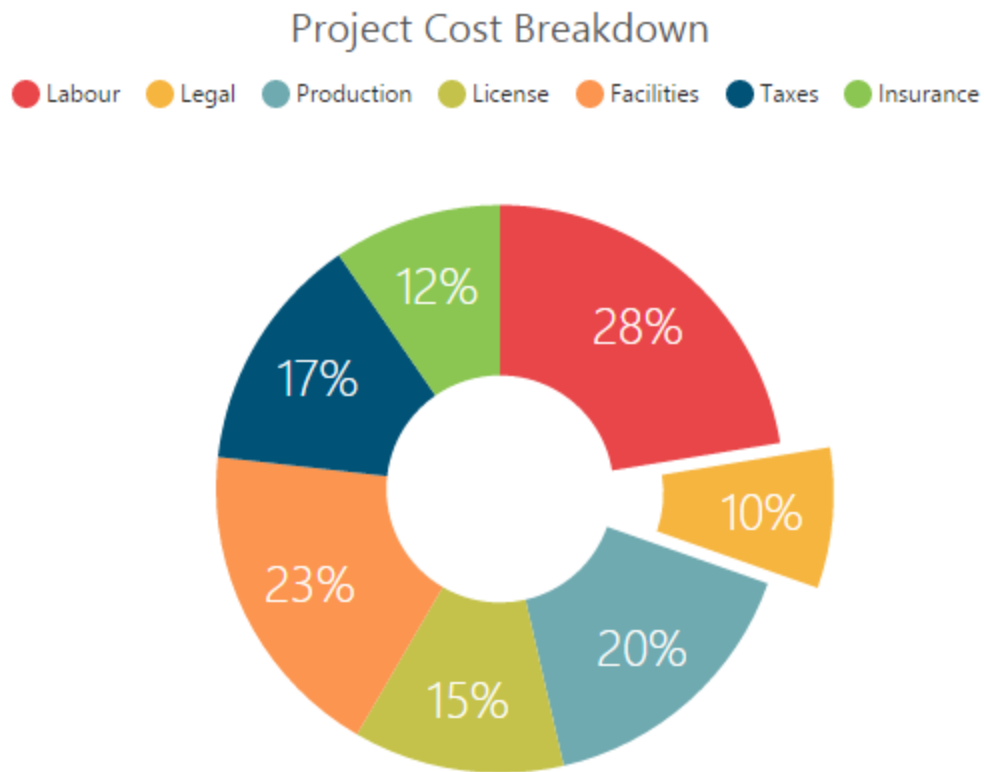
### *Explode a doughnut segment*

To explode a specific doughnut segment, set the index to be exploded by using the [explodeIndex](#) option of the series.

### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Set point index value to explode the doughnut segment.  
    explodeIndex: 1,  
    // ...  
  }],  
  // ...  
});
```



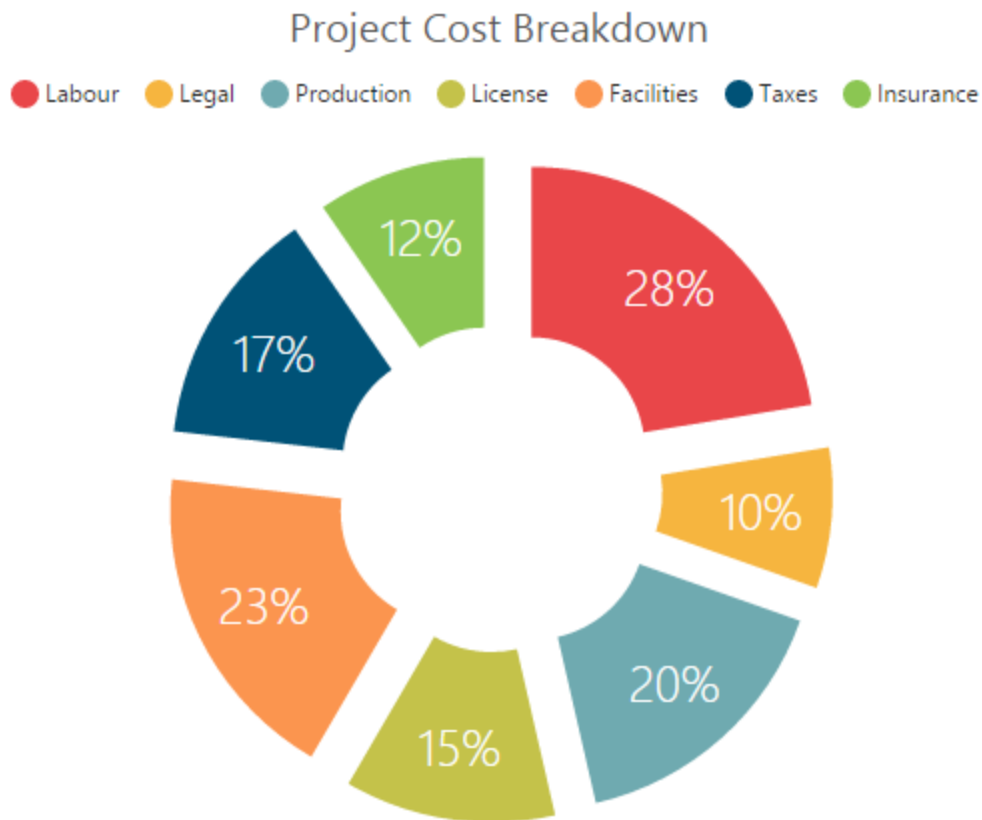


#### *Explode all the segments*

To explode all the segments, you can enable the [explodeAll](#) property of the series.

#### **JAVASCRIPT**

```
$("#chartContainer").ejChart({  
  // ...  
  series: [{  
    //Enable explodeAll property of doughnut chart  
    explodeAll: true,  
    // ...  
  }],  
  // ...  
});
```

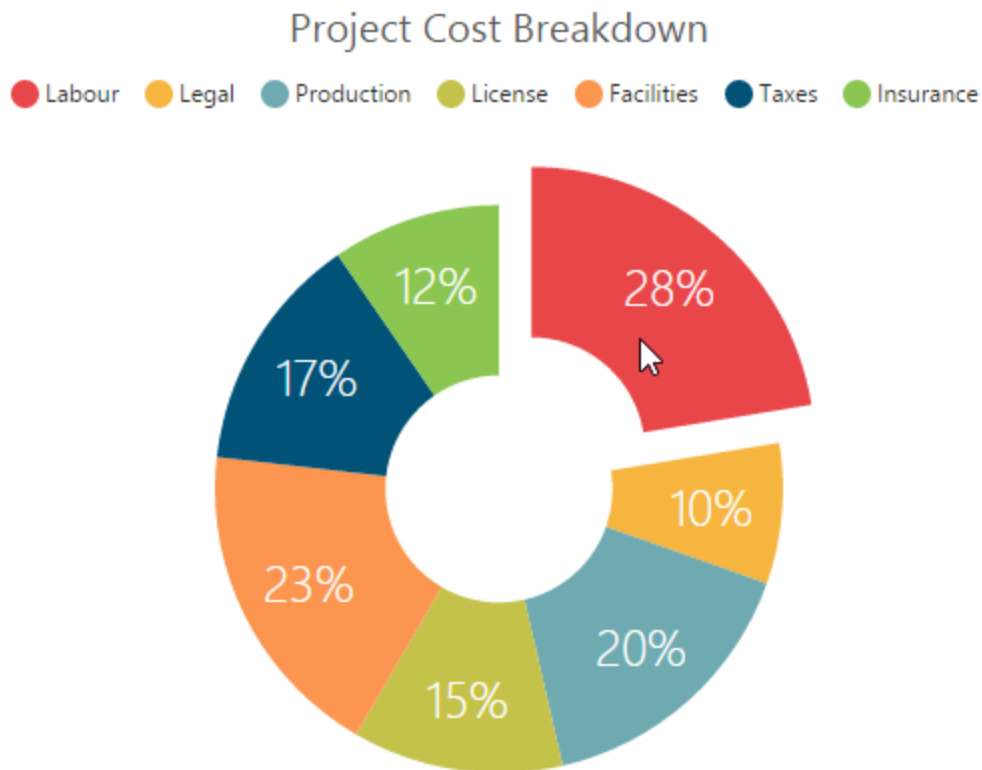


*Explode a doughnut segment on mouse over*

To explode a doughnut segment on a mouse over, you can enable the [explode](#) property of the series.

#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Enable doughnut explode option on mouse over the chart  
    explode: true,  
    // ...  
  }],  
  // ...  
});
```



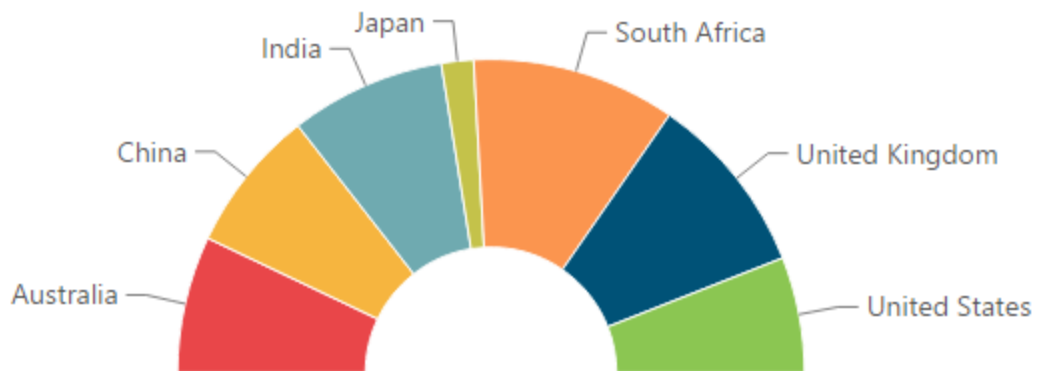
#### Sector of Doughnut

EjChart allows you to render all the data points/segments in the semi-doughnut, quarter- doughnut or in any sector by using the [startAngle](#) and [endAngle](#) options.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    type: 'doughnut',  
    //Set startAngle and endAngle to draw the semi pie chart  
    startAngle: -90, endAngle: 90  
    // ...  
  }],  
  // ...  
});
```

## Agricultural land in 2011 (% of land area)



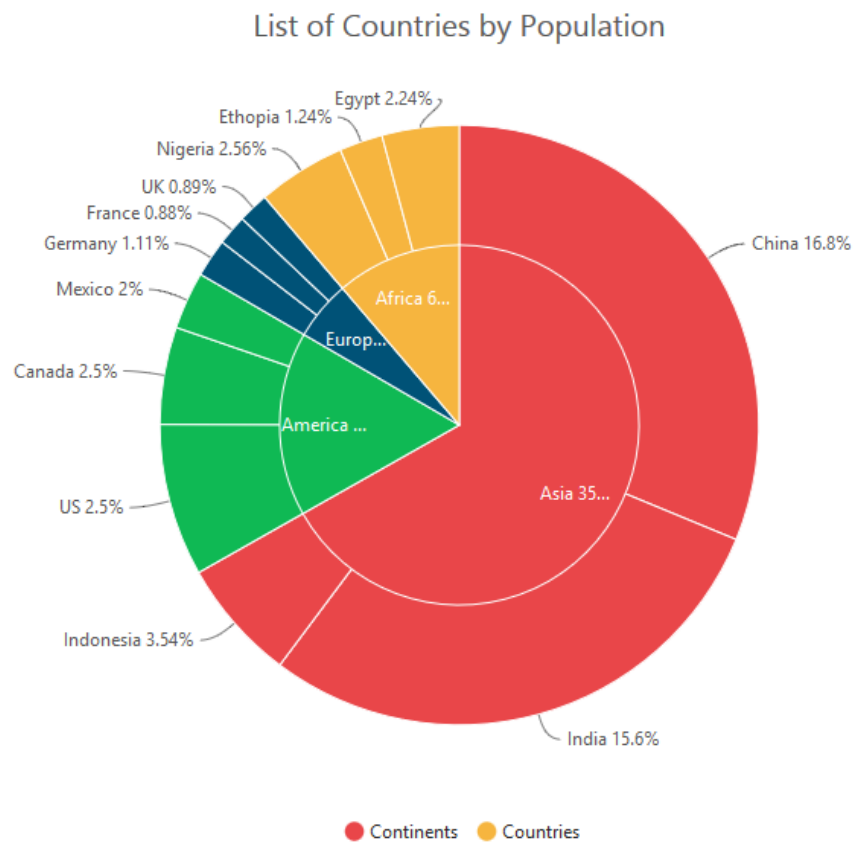
### Multiple Pie Chart

EjChart provides support to render more than one series in pie and in doughnut chart. Radius of each series is calculated based on the radius of the previous series. And in addition legend is displayed according to the list of chart series.

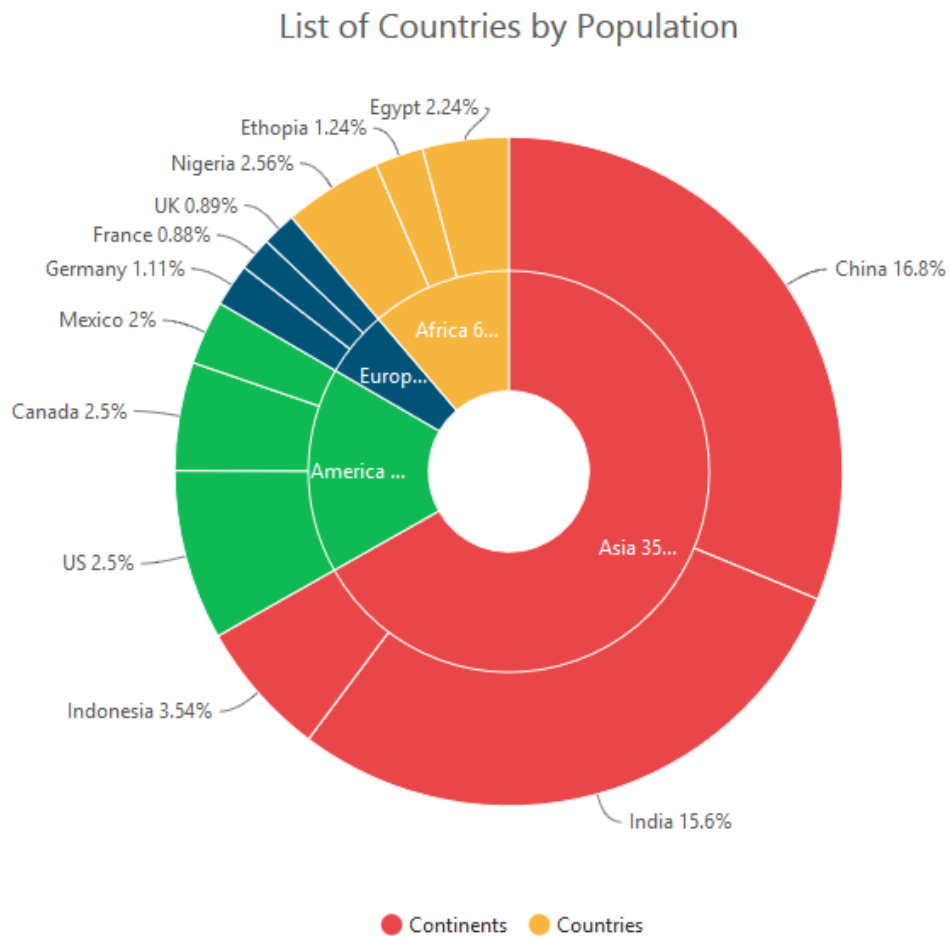
#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series:[{  
    //Adding multiple pie series  
    type: "pie",  
    //...  
  }, {  
    //Adding multiple pie series  
    type: "pie",  
    //...  
  }],  
  // ...  
});
```

### Multiple Pie



### Multiple Doughnut

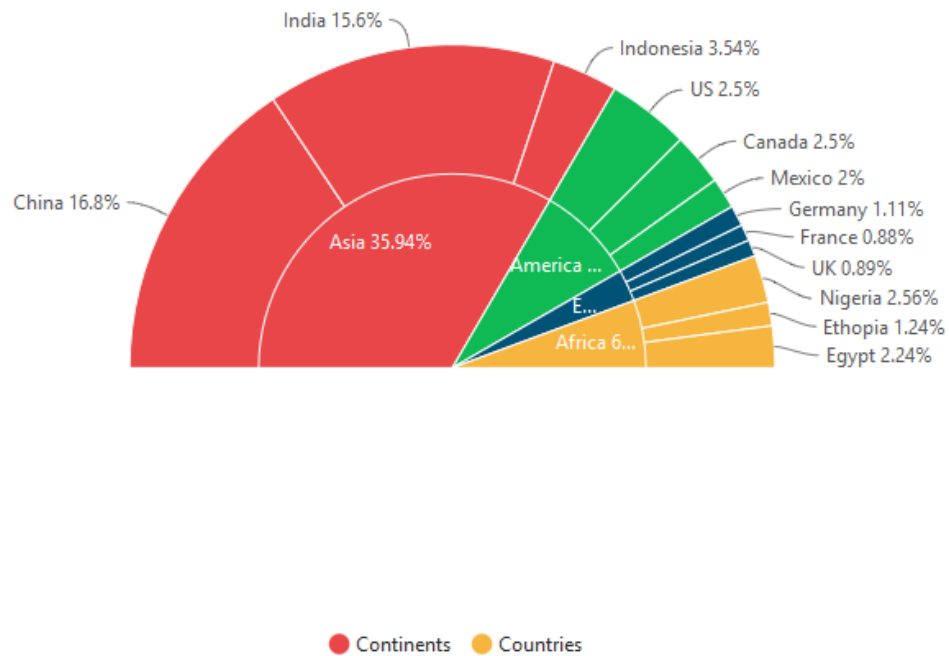


#### *Start and End Angle Support*

In the Multiple Pie chart, the start and end angle property is also supported.

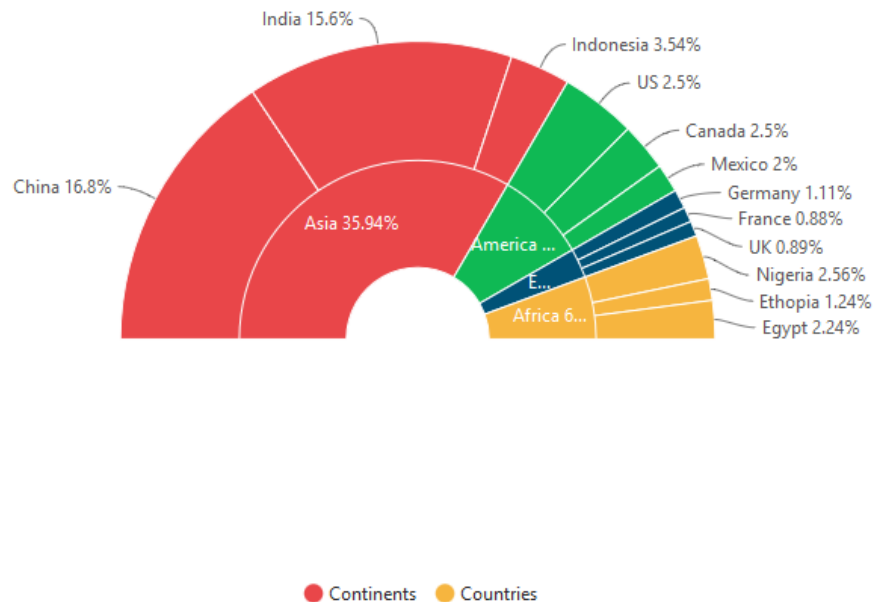
#### **Sector of Multiple Pie**

## List of Countries by Population



## Sector of Multiple Doughnut

## List of Countries by Population



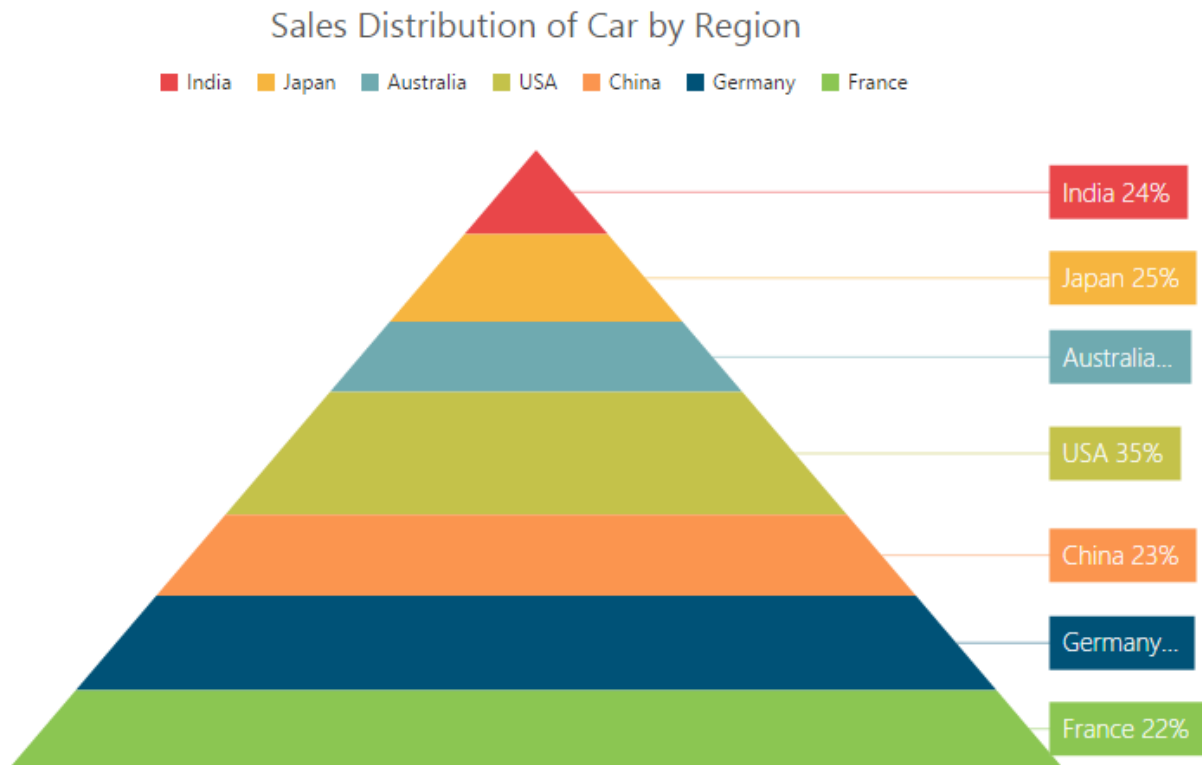
## Pyramid Chart

To create a Pyramid chart, you can specify the series [type](#) as “**pyramid**” in the chart series.

**JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  //...
  series: [{
    //Set chart type to series
    type: 'pyramid',
    // ...
  }],
  // ...
});
```



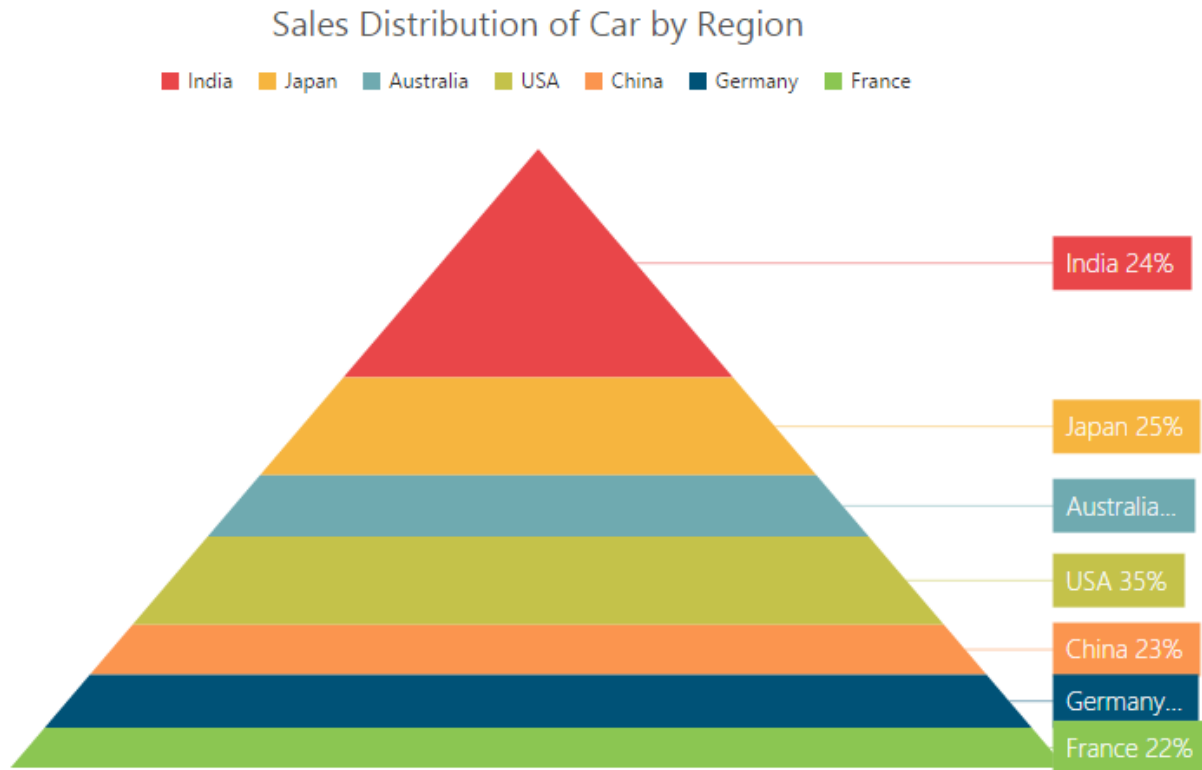


#### Pyramid Mode

Pyramid mode has two types, *linear* and *surface* respectively. The default “**pyramidMode**” type is “linear”.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  //...  
  series: [{  
    //Change pyramid mode  
    pyramidMode: 'surface',  
    // ...  
  }],  
  // ...  
});
```

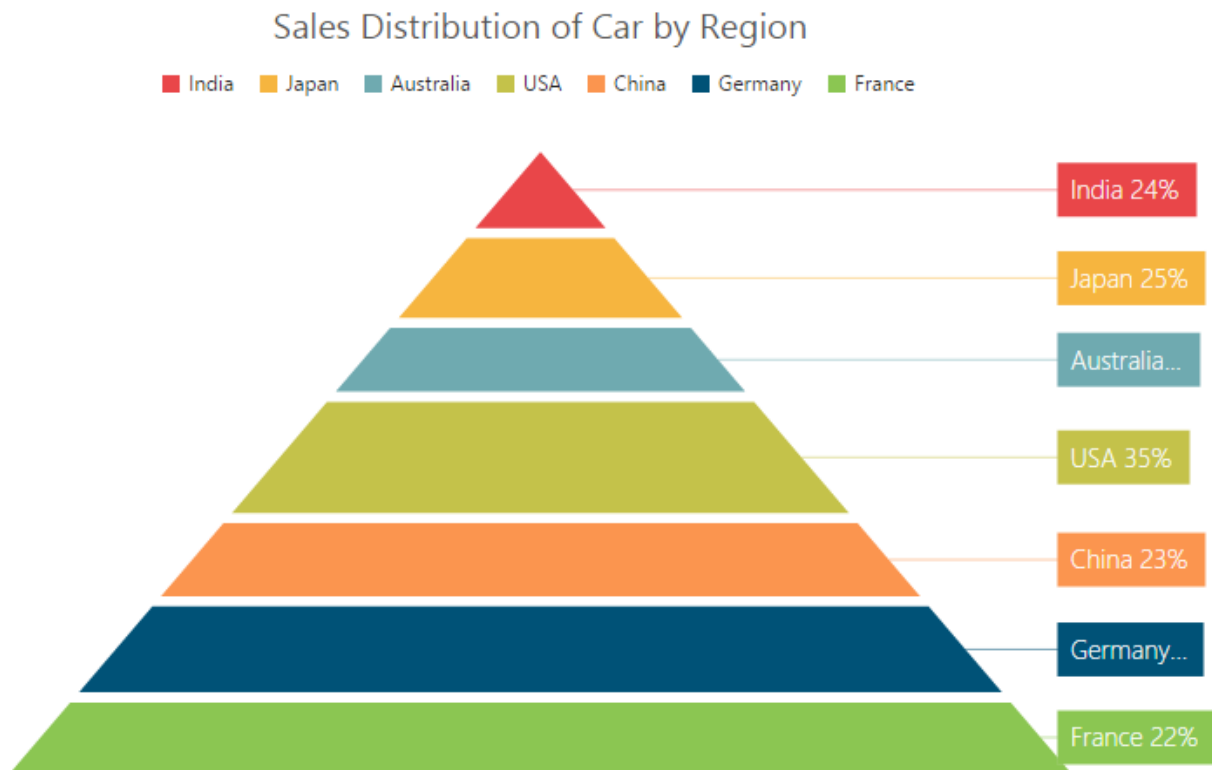


#### *Gap between the segments*

You can control the gap between the segments by using the [gapRatio](#) option of the series. Its ranges from 0 to 1.

#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  //...  
  series: [{  
    //Set gapRatio value to pyramid chart  
    gapRatio: 0.1,  
    // ...  
  }],  
  // ...  
});
```

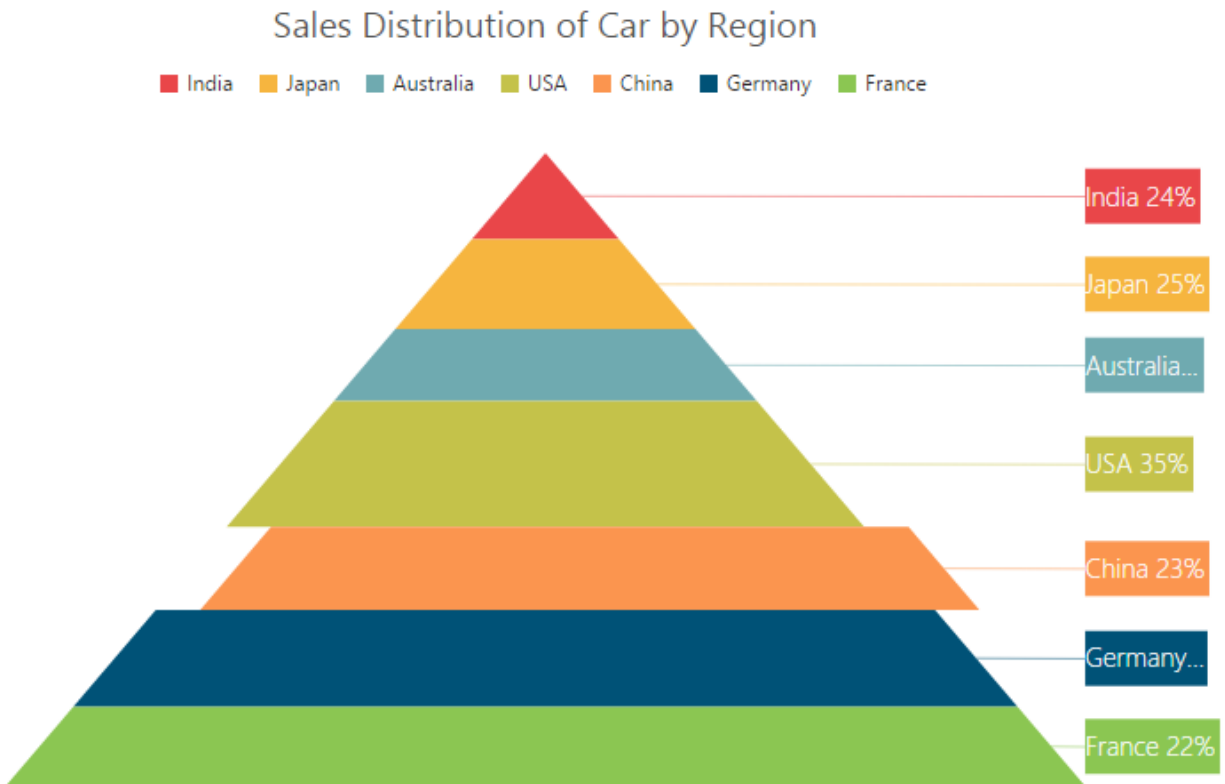


#### Explode a pyramid segment

You can explode a pyramid segment on the chart load by using the [explodeIndex](#) of the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  //...  
  series: [{  
    //Set point index value to explode the pyramid segment.  
    explodeIndex: 4,  
    // ...  
  }],  
  // ...  
});
```

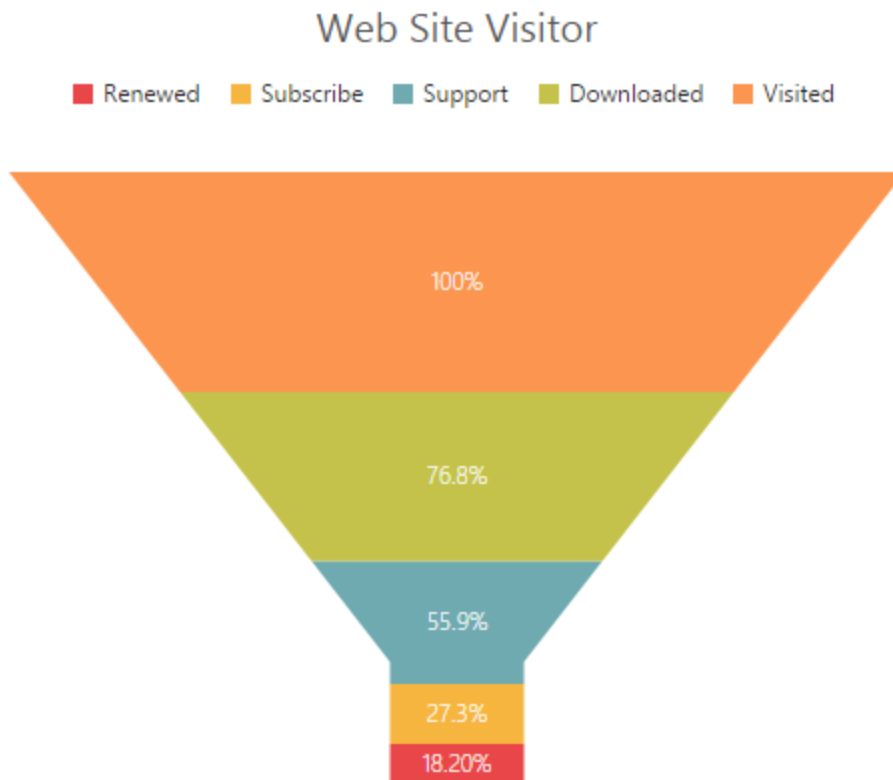


### Funnel Chart

You can create a funnel chart by setting the series [type](#) as “**funnel**” in the chart series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Set chart type to series  
    type: 'funnel',  
    // ...  
  }],  
  // ...  
});
```

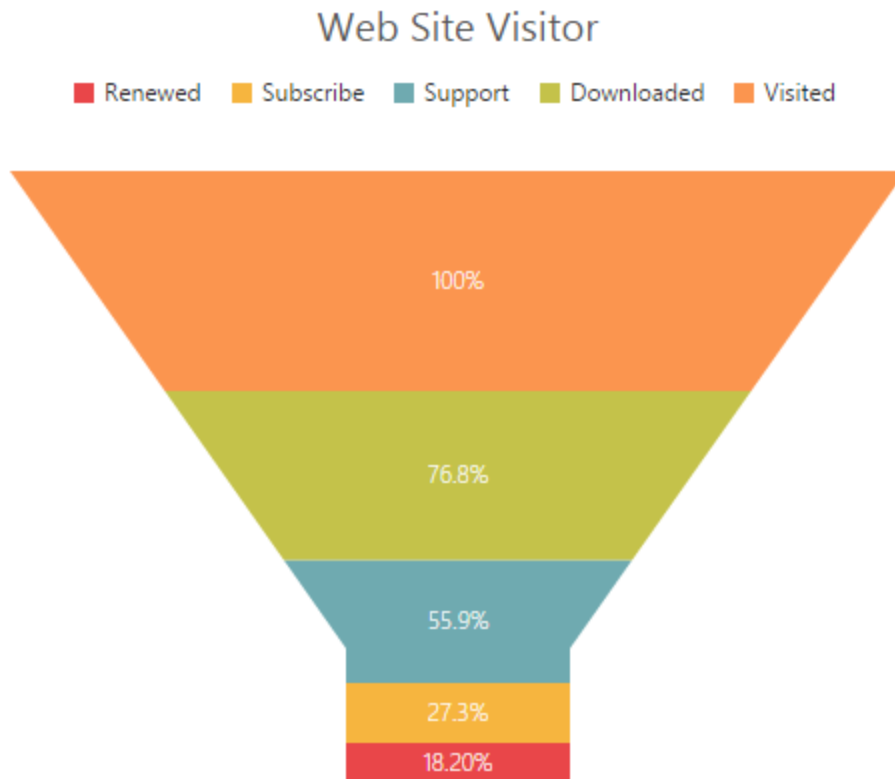


#### *Change the funnel width and height*

Funnel segments height and width is calculated from the chart size, by default. You can change this height and width directly without changing the chart size by using the [funnelHeight](#) and [funnelWidth](#) property of the series.

#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Change funnel height and width  
    funnelHeight:"22%",  
    funnelWidth:"25%",  
    // ...  
  }],  
  // ...  
});
```

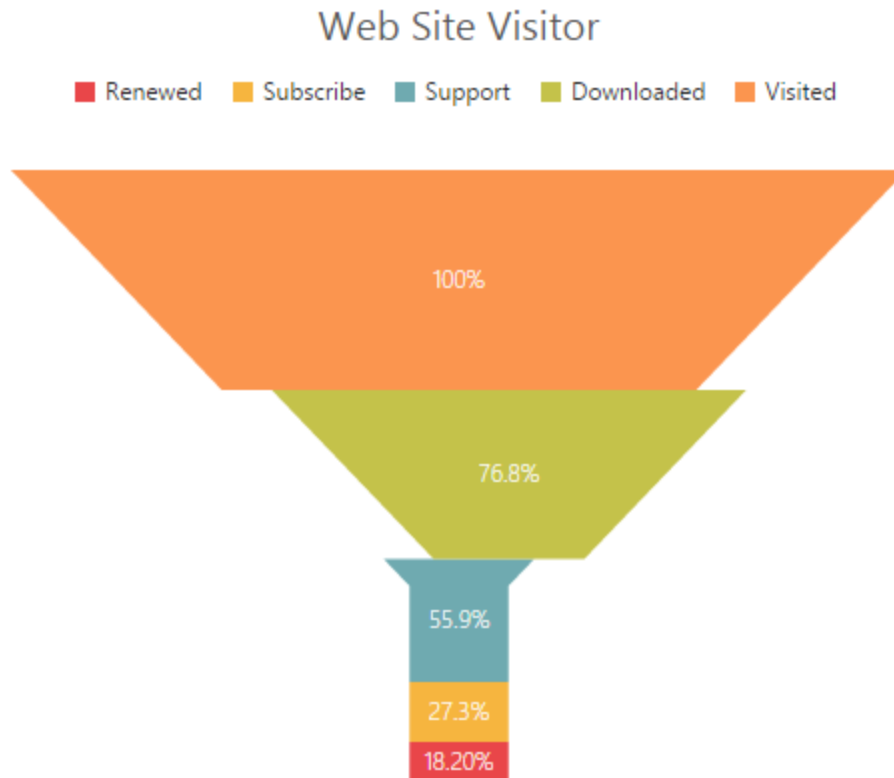


#### *Explode a funnel segment*

You can explode a funnel segment on the chart load by using the [explodeIndex](#) of the series.

#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Set point index value to explode the funnel segment.  
    explodeIndex: 3,  
    // ...  
  }],  
  // ...  
});
```

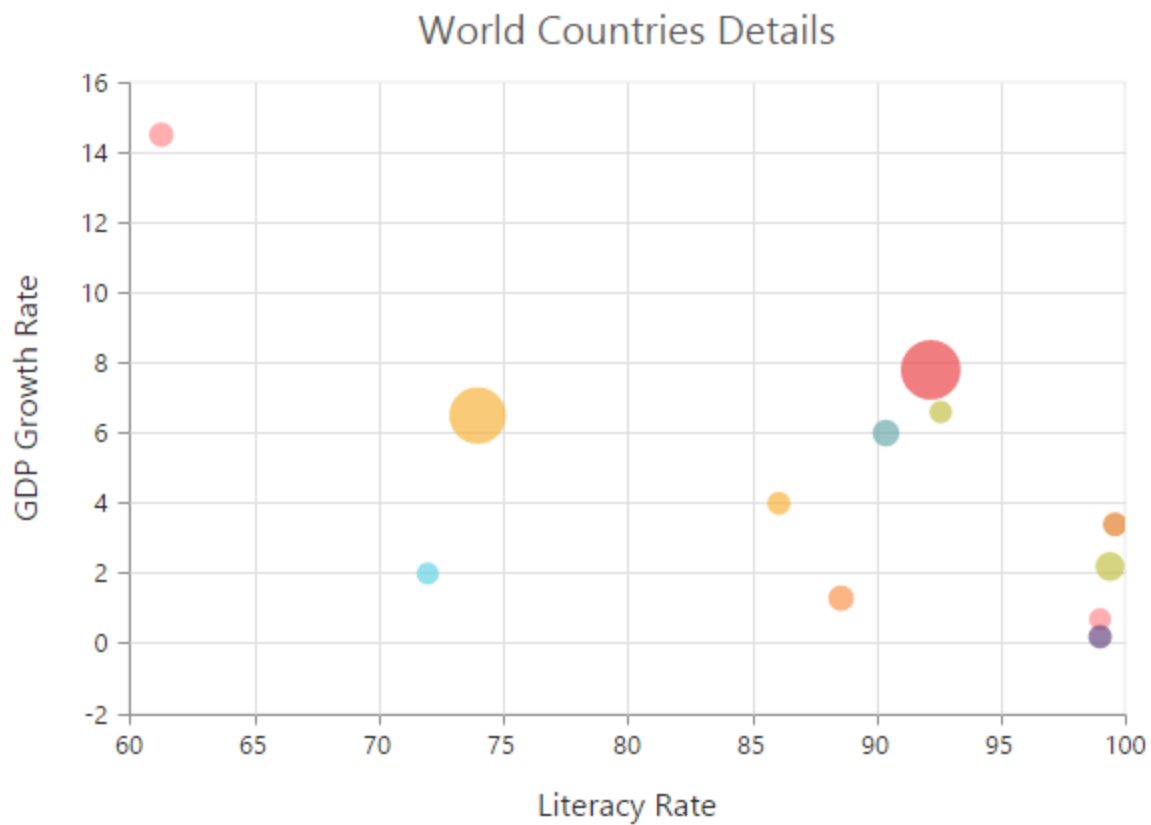


### Bubble Chart

To create a Bubble chart, you can set the series [type](#) as “**bubble**” in the chart series. Bubble chart requires 3 fields (*x*, *y* and *size*) to plot a point. Here, **size** is used to specify the size of each bubble segment.

### JAVASCRIPT

```
var chartData = [
  { month: 'Jan', sales: 35, profit:1.34 },
  { month: 'Feb', sales: 28, profit:1.05 },
  { month: 'Mar', sales: 34, profit:0.45 },
  { month: 'Apr', sales: 32, profit:1.10 },
  // ...
];
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Set chart type to series
    type: 'bubble',
    //Add datasource and set xName, yName and size to bubble chart
    dataSource: chartData,
    xName: "month",
    yName: "sales",
    size: "profit" ,
  }],
  // ...
});
```



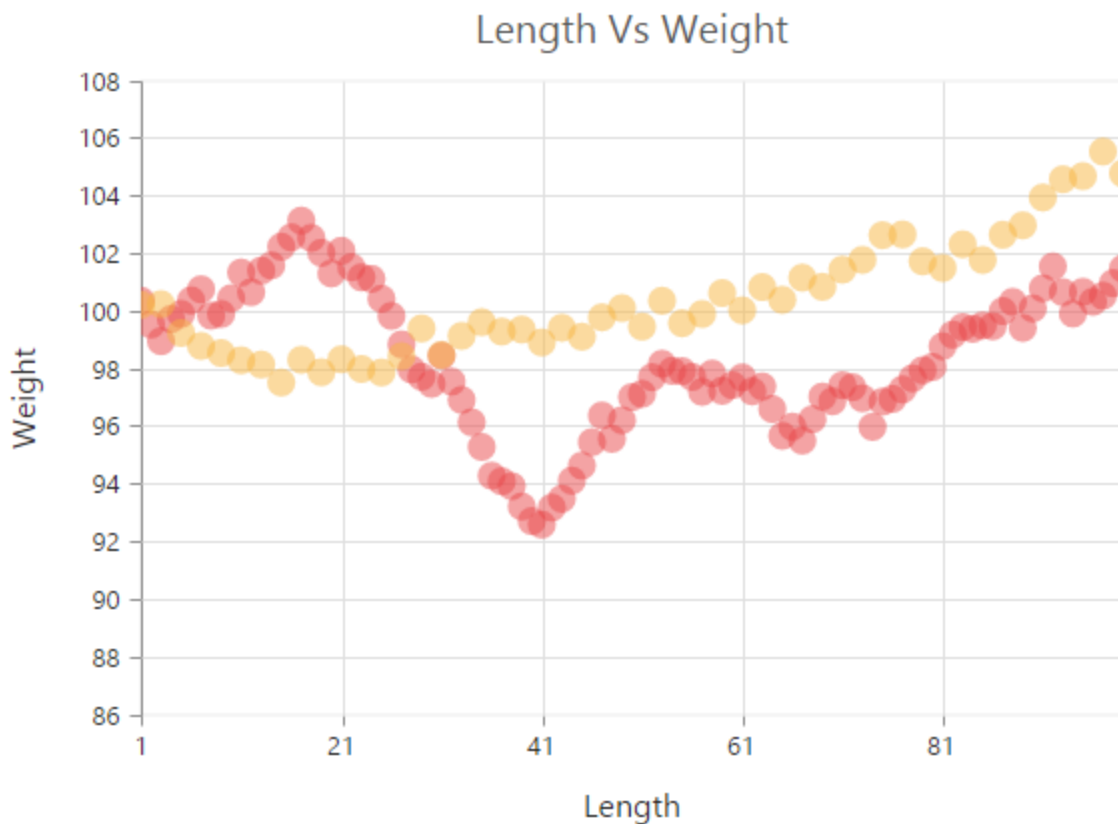
## Scatter

To create a Scatter chart, you can set the series [type](#) as “**scatter**” in the chart series.

## JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Set chart type to series  
    type: 'scatter',  
    // ...  
  }],  
  // ...  
});
```



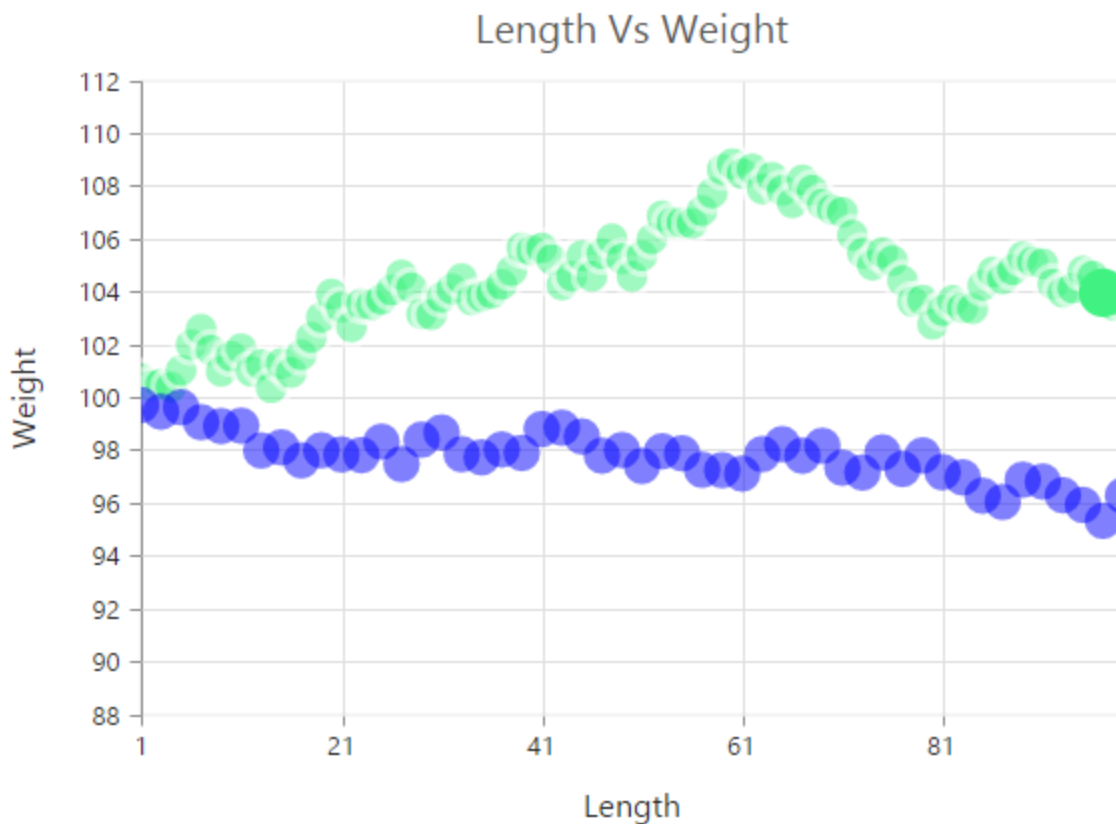


#### Customize the scatter chart

You can change the scatter size by using the [size](#) property of the series marker. And you can change the scatter color by using the series [fill](#) property.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    // ...  
    marker: {  
      //Change the scatter size  
      size: { height: 13, width: 13 }  
    },  
    //Set fill color to scatter series  
    fill: "#41F282",  
    // ...  
  }],  
  // ...  
});
```



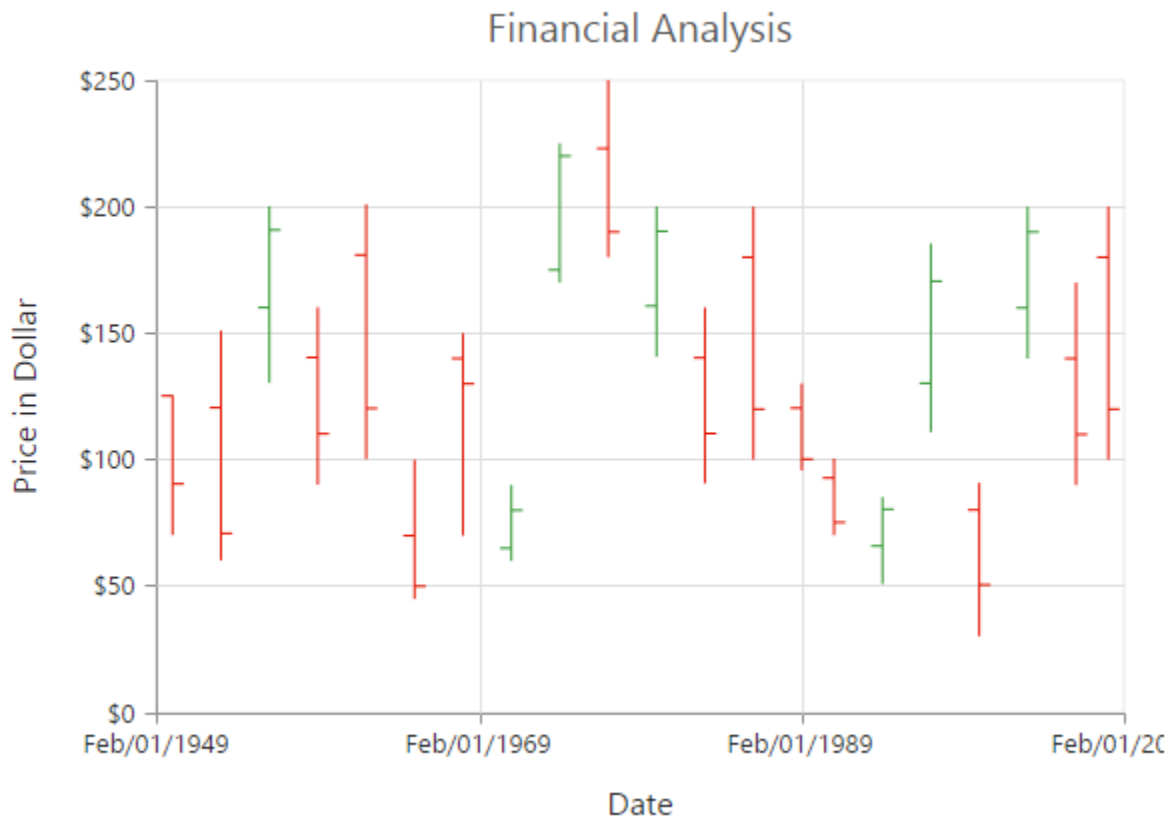
#### HiLoOpenClose Chart

To create a HiLoOpenClose chart, you can set the series [type](#) as “**hiloOpenClose**” in the chart series. HiLoOpenClose chart requires 5 fields ([x](#), [high](#), [low](#), [open](#) and [close](#)) to plot a segment.

#### JAVASCRIPT

```
var chartData = [
{ month: 'Jan', high: 38, low: 10, open: 38, close: 29 },
{ month: 'Feb', high: 28, low: 15, open: 18, close: 27 },
{ month: 'Mar', high: 54, low: 35, open: 38, close: 49 },
{ month: 'Apr', high: 52, low: 21, open: 35, close: 29 },
// ...
];
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
series: [{
//Set chart type to series
type: 'hiloOpenClose',
//Add datasource and set xName, high and low to hilo chart
dataSource: chartData,
xName: "month",
high: "high",
low: "low",
open: "open",
close: "close",
}],
// ...
}
```

```
});
```

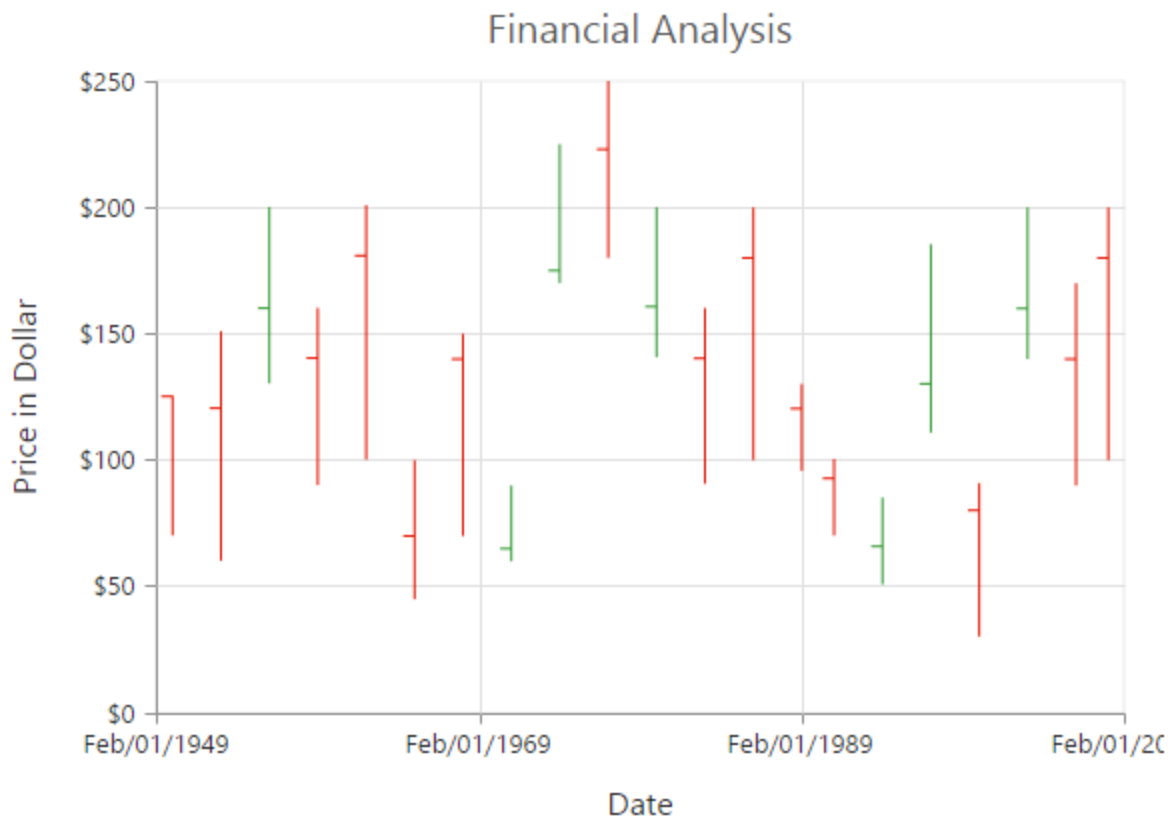


#### DrawMode

You can change the HiLoOpenClose chart [drawMode](#) to [open](#), [close](#) or *both*. The default value of [drawMode](#) is “both”.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // ...
    //Change the OHLC drawMode type
    drawMode: 'open',
  }],
  // ...
});
```

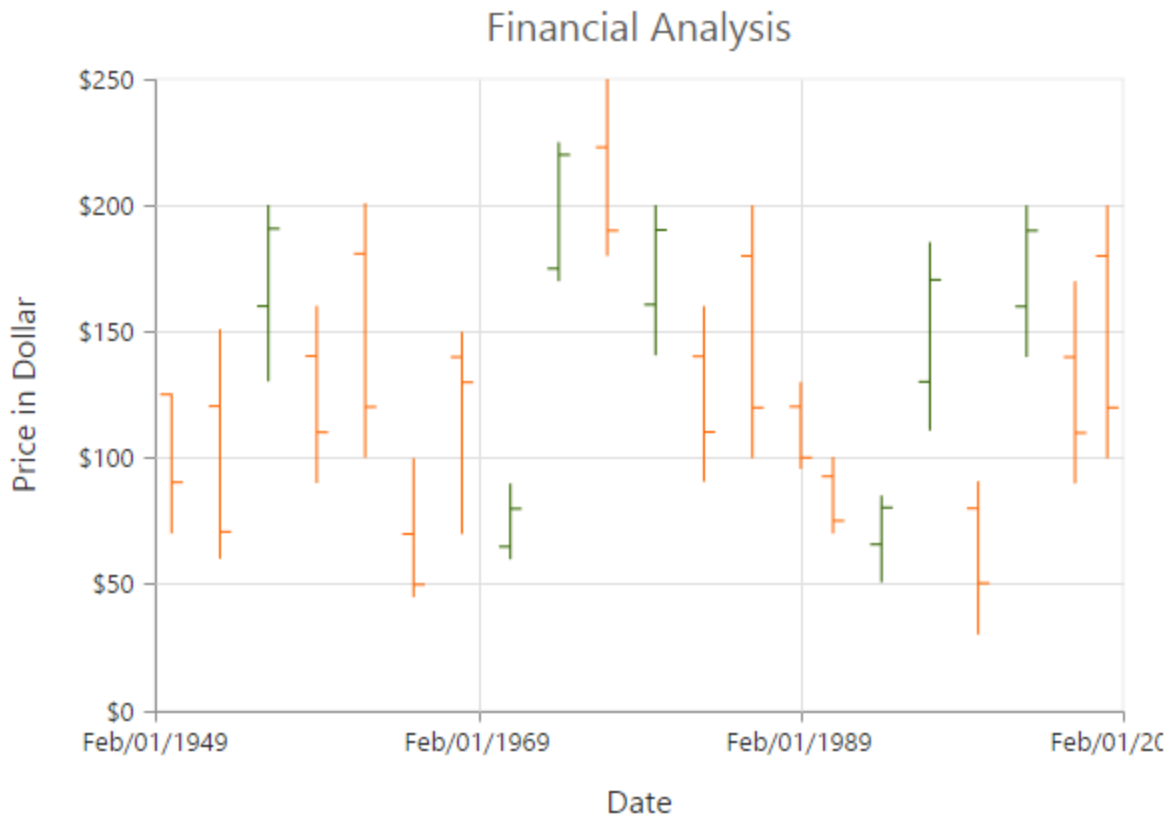


### Bull and Bear Color

HiLoOpenClose chart [bullFillColor](#) is used to specify a fill color for the segments that indicates an increase in stock price in the measured time interval and [bearFillColor](#) is used to specify a fill color for the segments that indicates a decrease in the stock price in the measured time interval.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Change bullFill and bearFill color of HiLoOpenClose chart
    bullFillColor: '#FF6600',
    bearFillColor: '#336600',
    // ...
  }],
  // ...
});
```



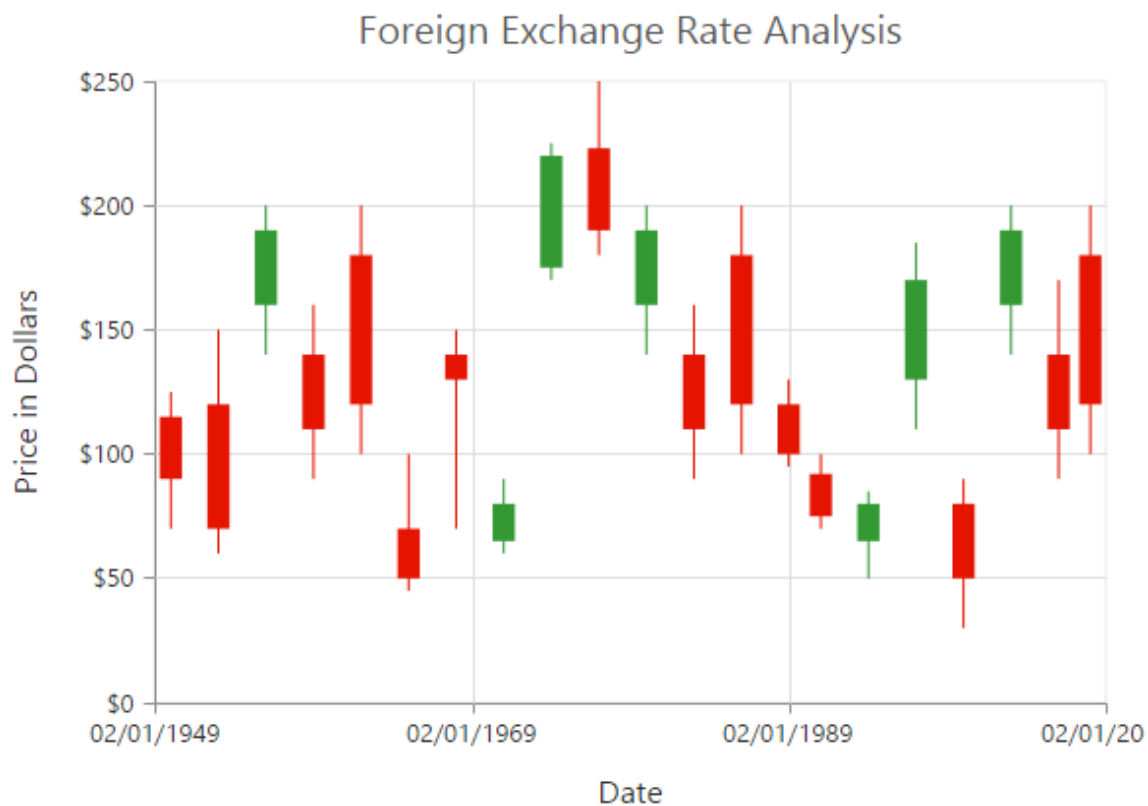
## Candle

You can create a Candle chart by specifying the series [type](#) as “**candle**” in the chart series. Candle chart requires 5 fields ([x](#), [high](#), [low](#), [open](#) and [close](#)) to plot a segment.

## JAVASCRIPT

```
var chartData = [
{ month: 'Jan', high: 38, low: 10, open: 38, close: 29 },
{ month: 'Feb', high: 28, low: 15, open: 18, close: 27 },
{ month: 'Mar', high: 54, low: 35, open: 38, close: 49 },
{ month: 'Apr', high: 52, low: 21, open: 35, close: 29 },
// .....
];
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
series: [{
//Set chart type to series
type: 'candle',
//Add datasource and set xName, high and low to hilo chart
dataSource: chartData,
xName: "month",
high: "high",
low: "low",
open: 'open',
close: 'close'
}],
// ...
}
```

```
});
```

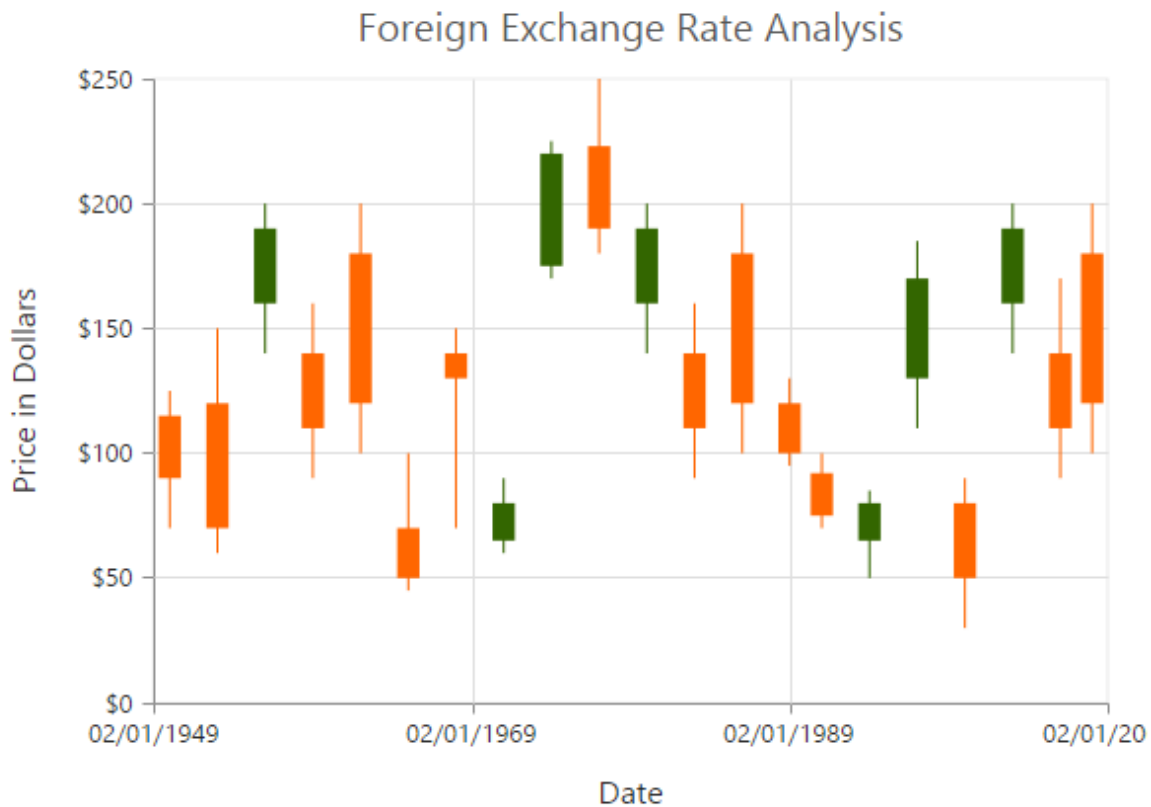


#### Bull and Bear Color

Candle chart [bullFillColor](#) is used to specify a fill color for the segments that indicates an increase in the stock price in the measured time interval and [bearFillColor](#) is used to specify a fill color for the segments that indicates a decrease in the stock price in the measured time interval.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Change bullFill and bearFill color of candle chart
    bullFillColor: '#FF6600',
    bearFillColor: '#336600',
    // ...
  }],
  // ...
});
```

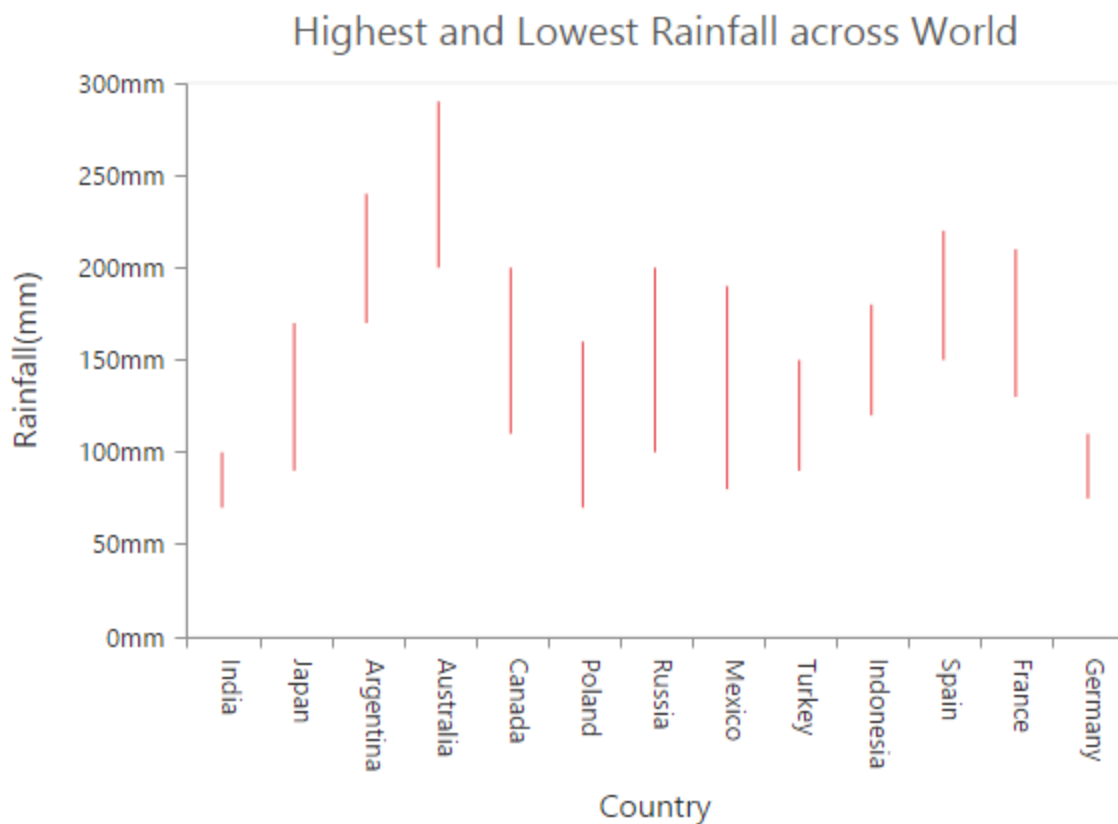


## HiLo

HiLo chart is created by setting the series [type](#) as “**hilo**” in the chart series. HiLo chart requires 3 fields ([x](#), [high](#) and [low](#)) to plot a segment.

## JAVASCRIPT

```
var chartData = [
  { month: 'Jan', high: 38, low: 34 },
  { month: 'Feb', high: 28, low: 15 },
  { month: 'Mar', high: 54, low: 45 },
  { month: 'Apr', high: 32, low: 21 },
  // ...
];
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Set chart type to series
    type: 'hilo',
    //Add datasource and set xName, high and low to hilo chart
    dataSource: chartData,
    xName: "month",
    high: "high",
    low: "low",
  }],
  // ...
});
```



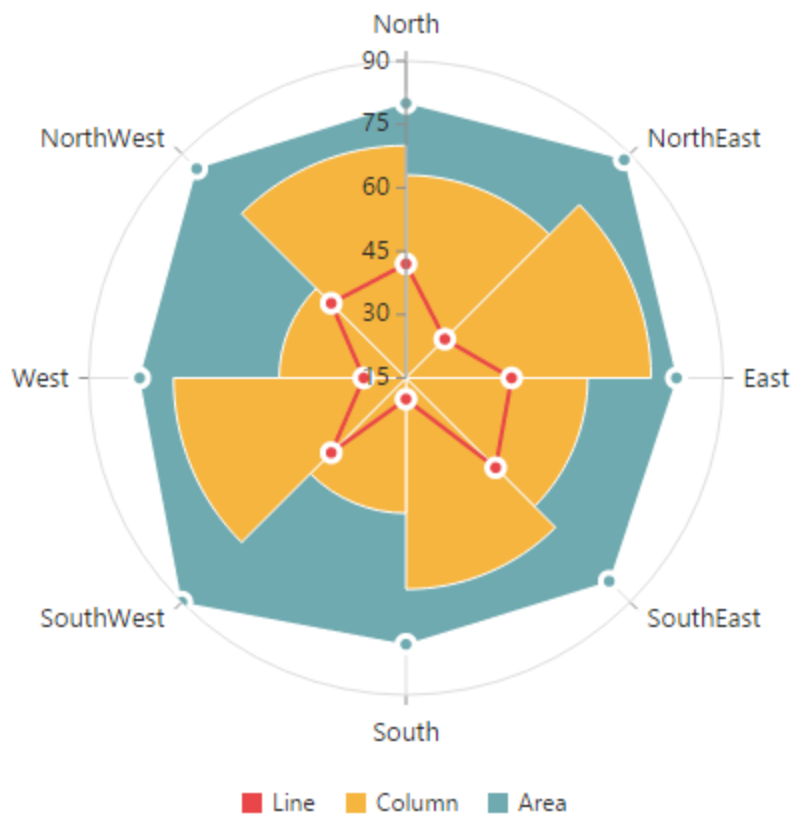
#### Polar

Polar chart is created by setting the series [type](#) as **polar** in the chart series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    //Set chart type to series  
    type: 'polar'  
  }],  
  // ...  
});
```



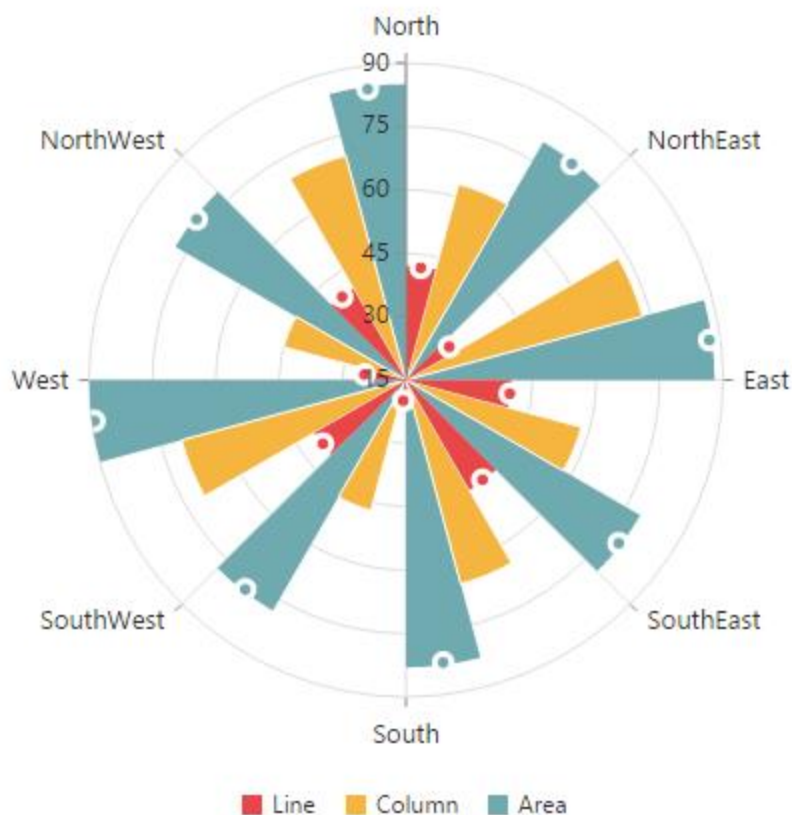


### DrawType

Polar **DrawType** property is used to change the series plotting type to *Line*, *scatter*, *rangeColumn*, *stackingArea*, *spline*, *Column* or *Area*. The default value of DrawType is **Line**.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Change polar series drawType
    drawType: 'column',
    // ...
  }],
  // ...
});
```

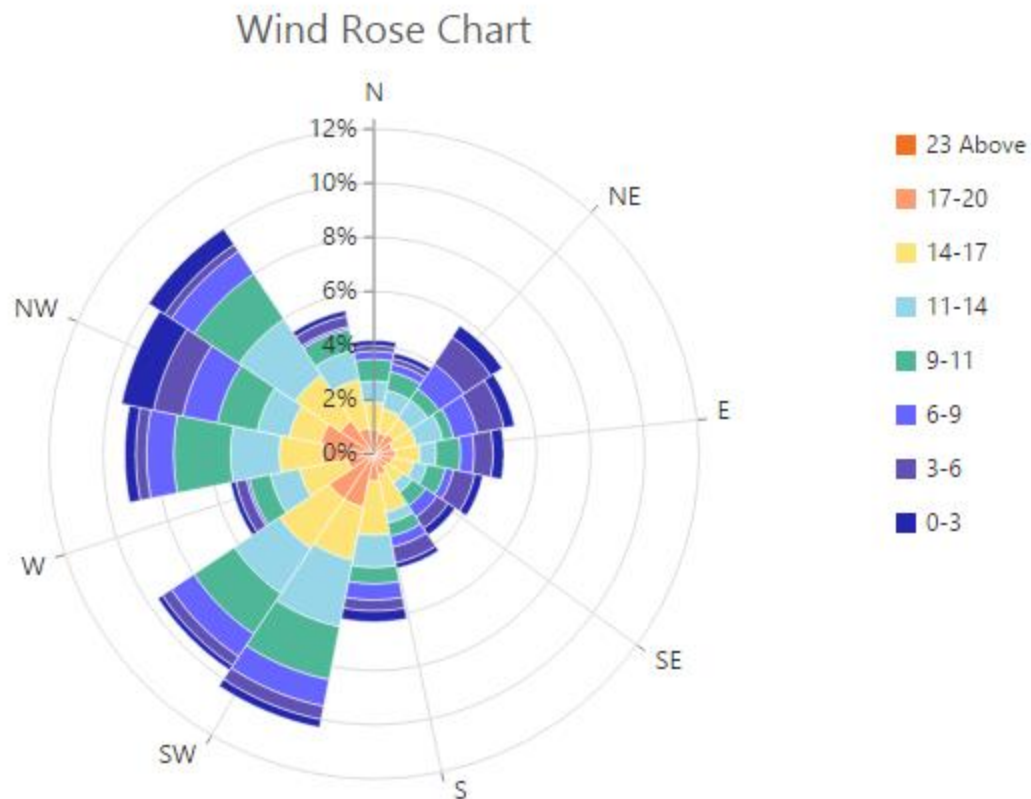


### Stack columns in Polar chart

By using the [isStacking](#) property, you can specify whether the column has to be stacked when the [drawType](#) is column. Its default value is **false**.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Enable isStacking property for stacked column polar chart
    isStacking: true
    // ...
  }],
  // ...
});
```

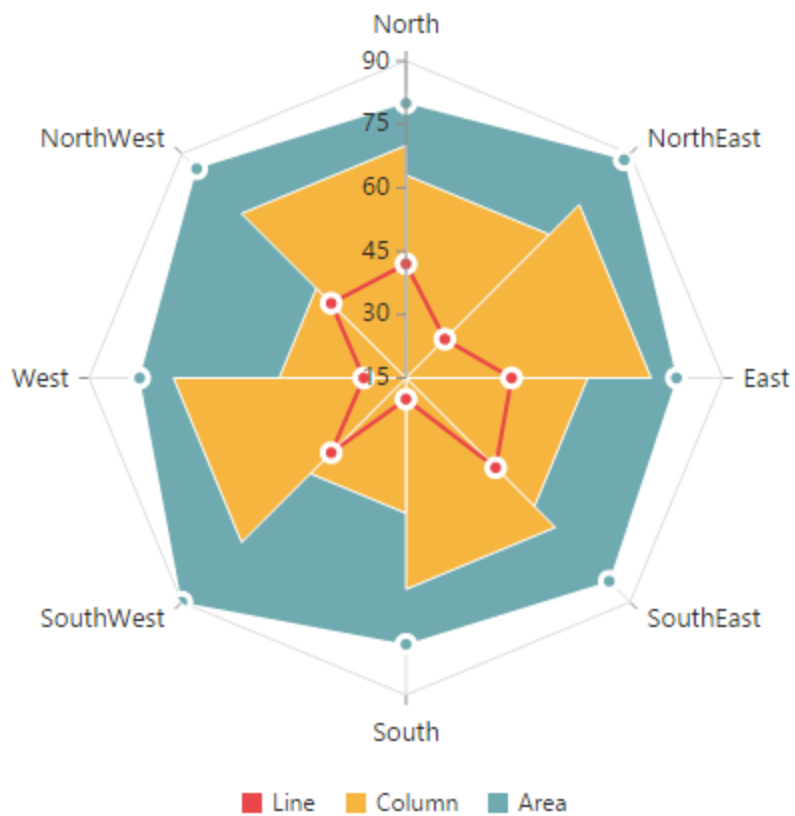


### Radar Chart

To create a Radar chart, you can specify the series [type](#) as “**radar**” in the chart series.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Set chart type to series
    type: 'radar',
    // ...
  }],
  // ...
});
```

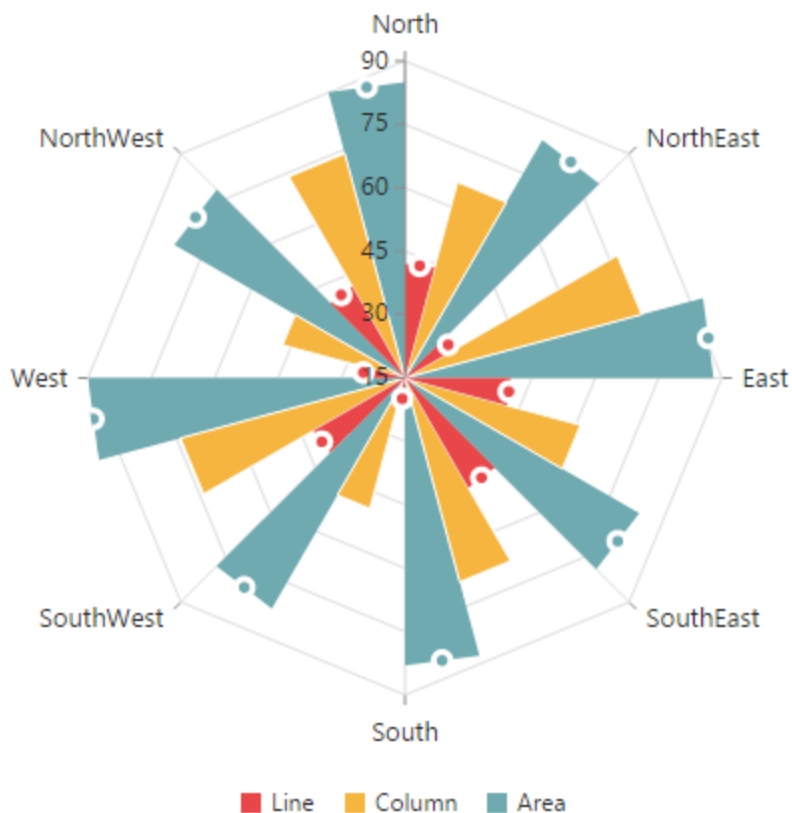


### DrawType

Radar **DrawType** property is used to change the series plotting type to *Line*, *scatter*, *rangeColumn*, *stackingArea*, *spline*, *Column* or *Area*. The default value of DrawType is **Line**.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Change radar series drawType
    drawType: 'column',
    // ...
  }],
  // ...
});
```

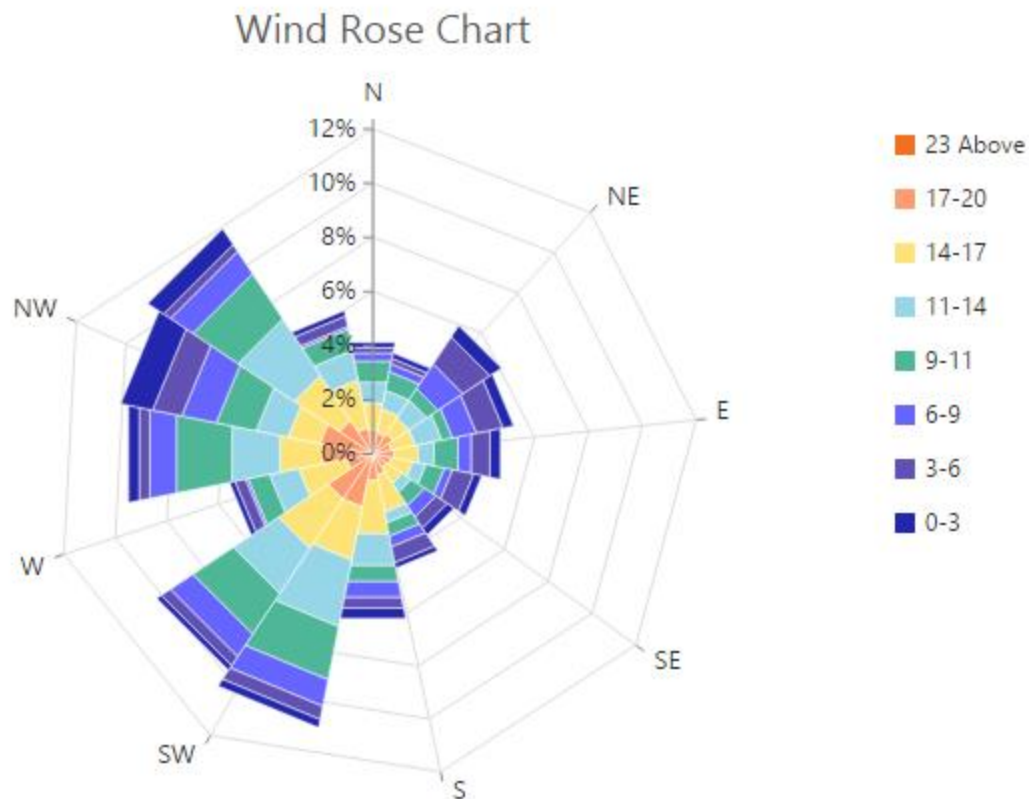


#### Stack columns in Radar chart

By using the [isStacking](#) property, you can specify whether the column has to be stacked when the [drawType](#) is set as *column*. Its default value is set to **false**.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Enable isStacking property for stacked column radar chart
    isStacking: true
    // ...
  }],
  // ...
});
```



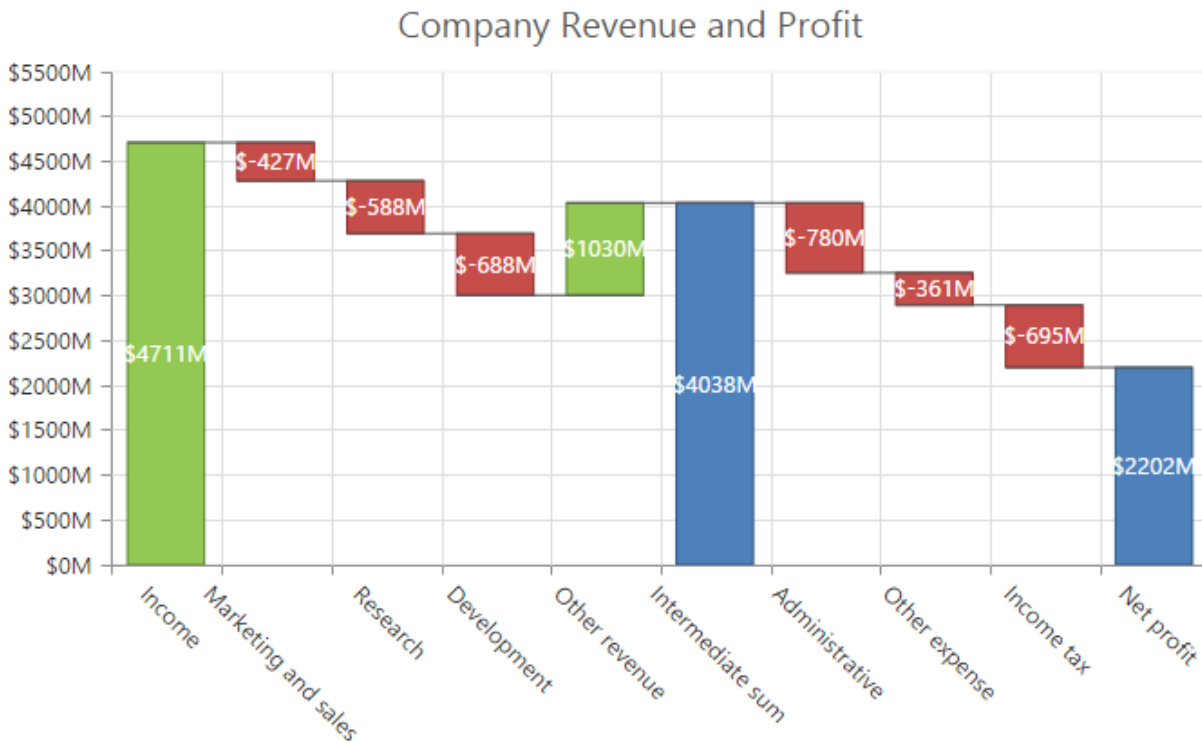
#### Waterfall Chart

For rendering a Waterfall chart, set series [type](#) as “**waterfall**” in the chart series. To change the waterfall series segment color use [fill](#) option of series and use [positiveFill](#) property to differentiate the positive segments.

**Note:** The inline property of the **series.positiveFill** has the first priority and override the **series.fill**.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Change type and color of the series.
    type: waterfall,
    fill: "#C64E4A",
    positiveFill: "#C64E4A"
    // ...
  }],
  // ...
});
```



### ShowIntermediateSum

To display the summary of values since the last intermediate point of the waterfall series, set **showIntermediateSum** property as true in the specific point.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    points: [
      //Enable showIntermediateSum in to a point.
      // ...
      { x: "Intermediate sum", showIntermediateSum: true }
      // ...
    ],
    // ...
  }],
  // ...
});
```

### ShowTotalSum

The sum of all previous point in the waterfall series is displayed on enabling the **showTotalSum** property for a specific point.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
```

```

points: [
  //Enable showTotalSum in to a point.
  // ...
  { x: "Total sum", showTotalSum: true }
  // ...
],
// ...
}],
// ...
});

```

### ConnectorLine

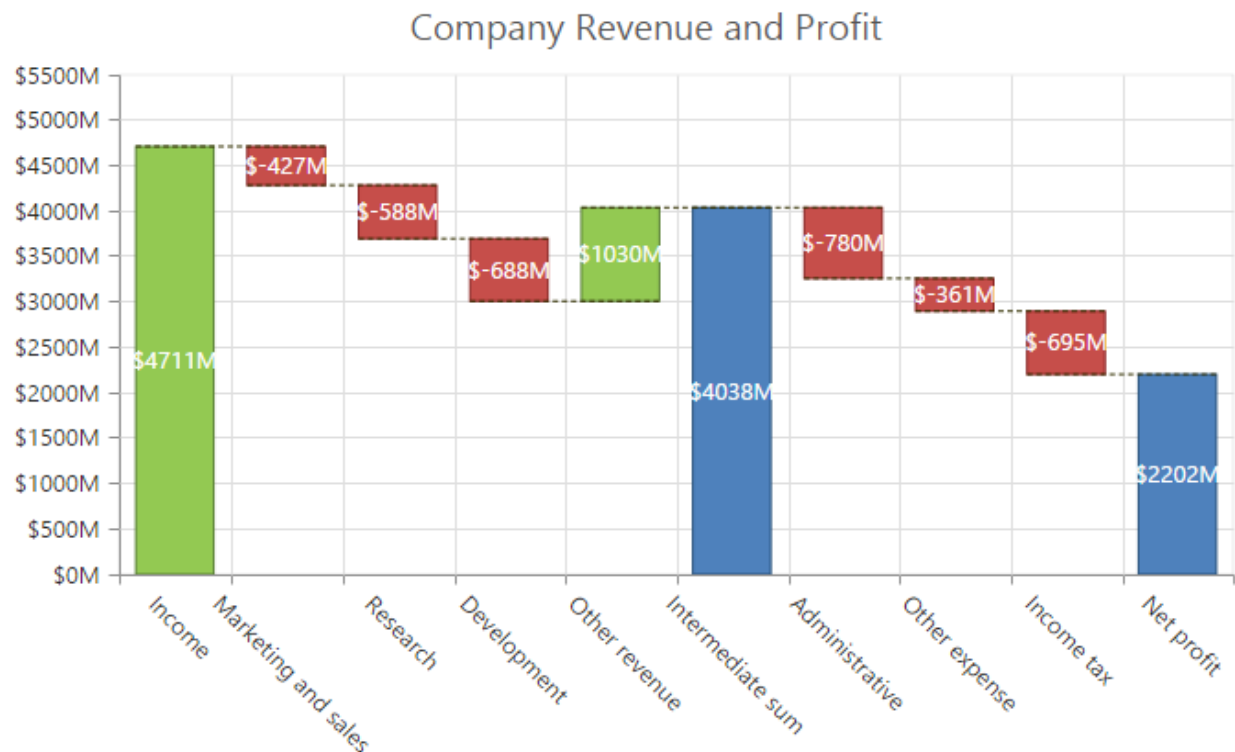
To customize the connector line color, width, opacity and dashArray of the waterfall series, you can use [connectorLine](#) option of series.

### JAVASCRIPT

```

var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //customize waterfall series connector line styles
    connectorLine: {color: "#333000", width: 1, opacity: 1, dashArray: "3,2"},
    // ...
  }],
  // ...
});

```



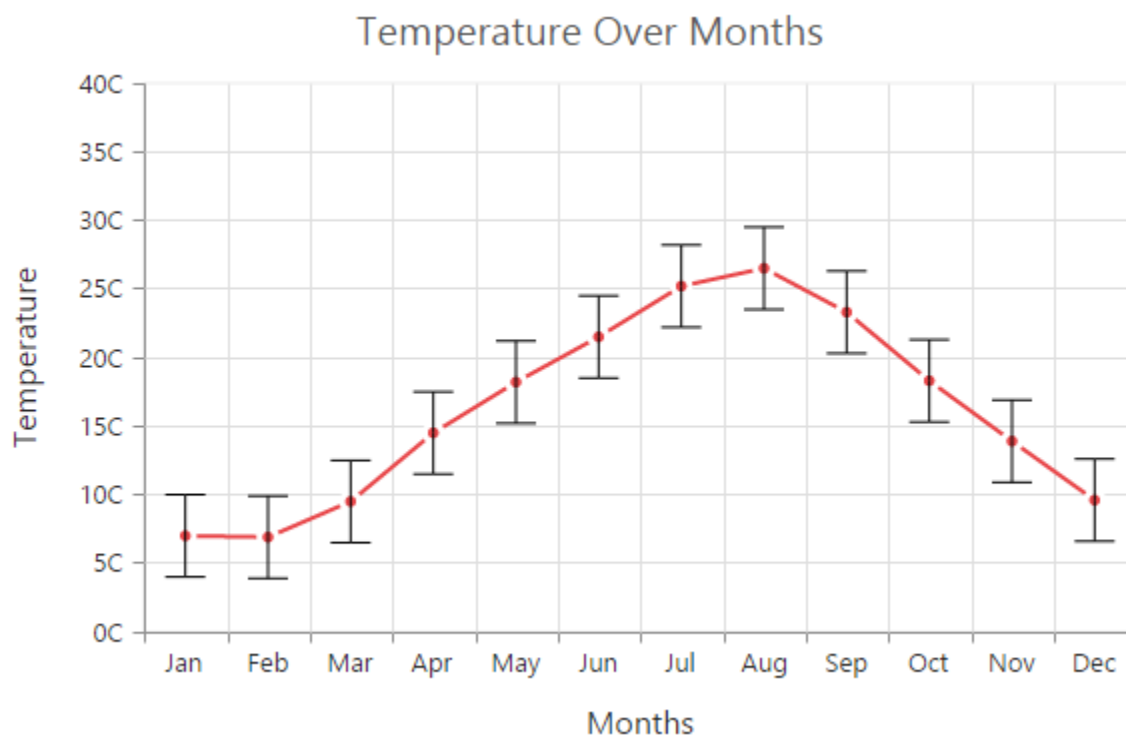


### Error bar Chart

EjChart can generate Error bar for Cartesian type series (*Line, Column, Bar, Scatter, Area, Candle, HiLo, etc.*). To render the Error bar for the series, set [visibility](#) as "visible" to [errorBar](#) in the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    //...
    //To toggle the error bar visibility
    errorBar: {
      visibility: "visible"
    }
  }]
  // ...
});
```



### Changing Error Bar Type

You can change the error bar rendering type using [type](#) (like *fixedValue, percentage, standardDeviation, standardError and custom*) option of errorBar. To change the error bar line length you can use [verticalErrorValue](#) property.

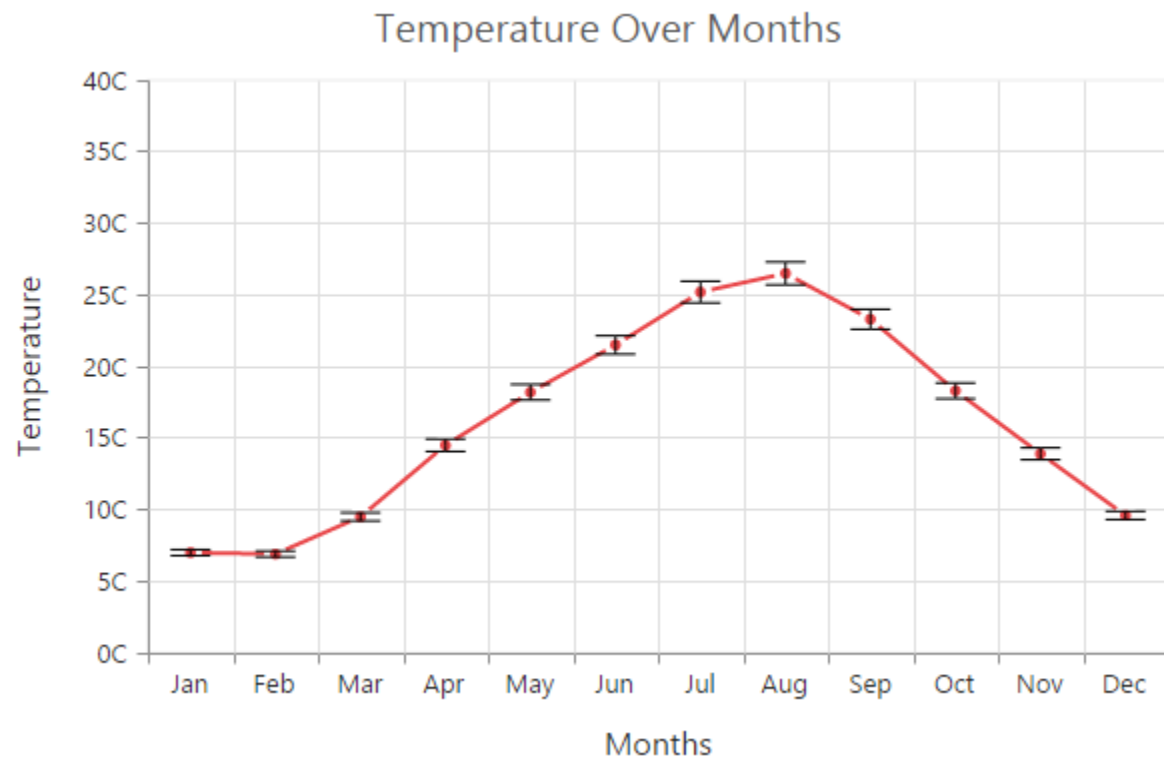
#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    //...
    //Change the error bar type
  }]
});
```

```

errorBar: {
  type: "percentage",
  verticalErrorValue:3
}
}]
// ...
});

```



#### Customizing error bar type

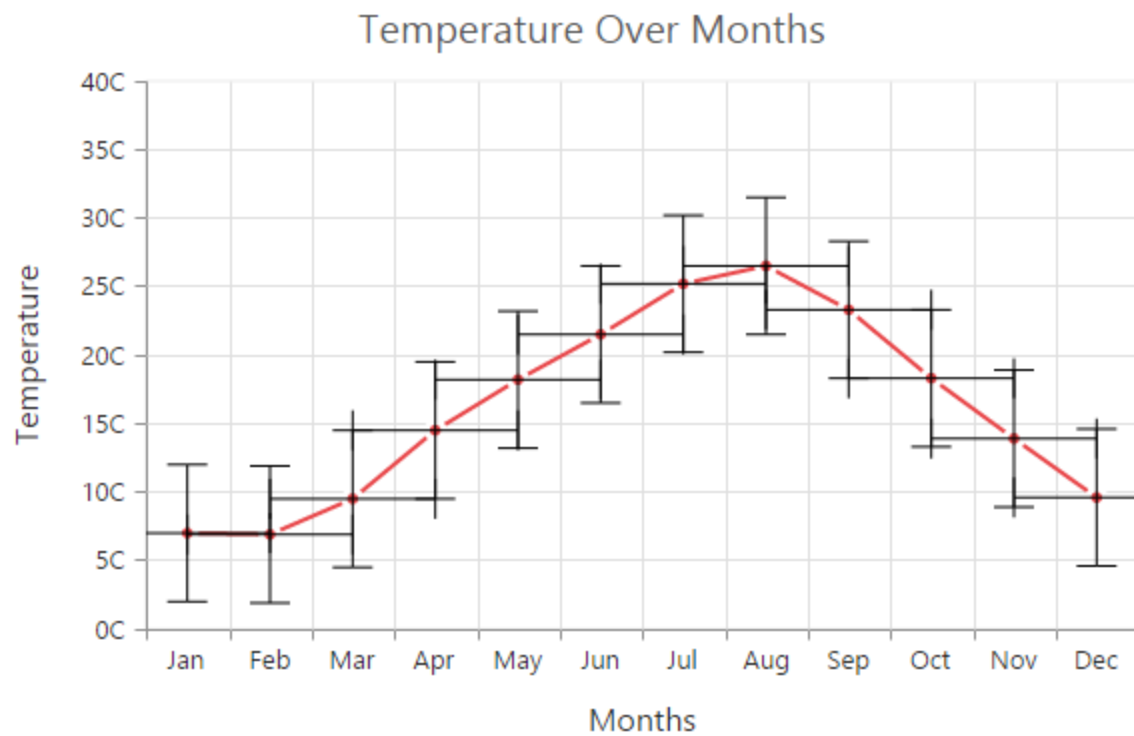
To customize the error bar type, set error bar [type](#) as “**custom**” and then change the horizontal/vertical positive and negative value of error bar.

#### JAVASCRIPT

```

var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    //...
    //Change the error bar type
    errorBar: {
      type: "custom",
      verticalPositiveErrorValue:5,
      horizontalPositiveErrorValue:1,
      verticalNegativeErrorValue:5,
      horizontalNegativeErrorValue:1
    }
  }]
  // ...
});

```

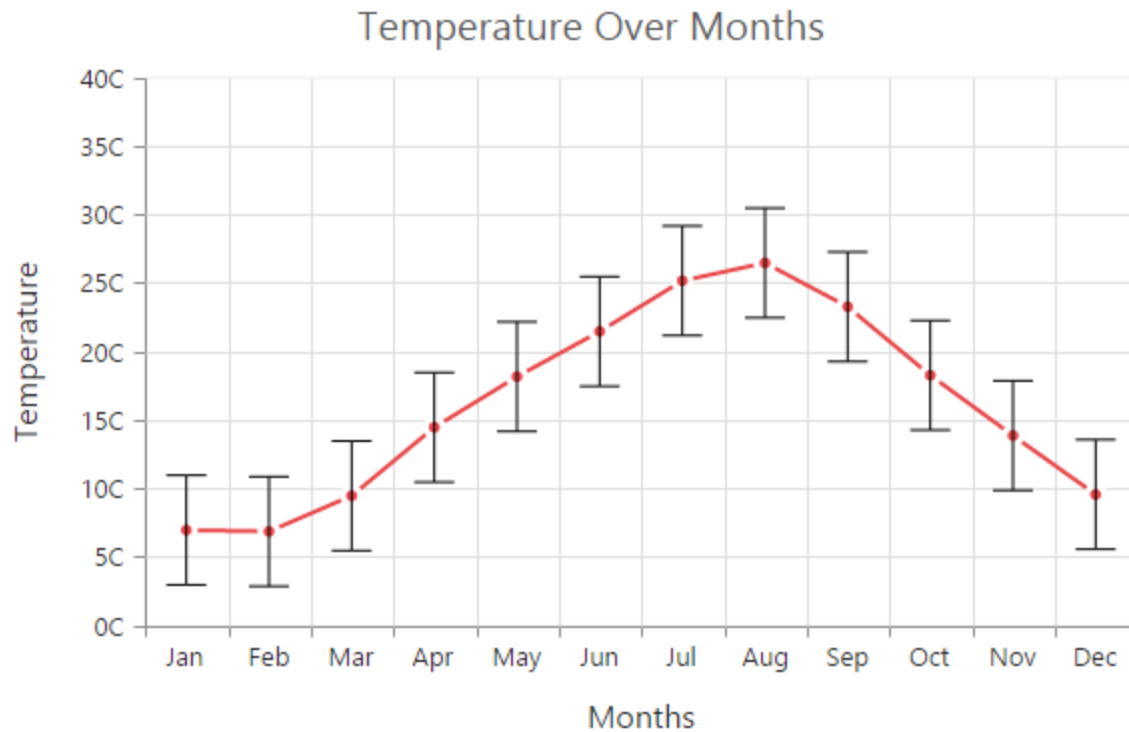


#### Changing Error Bar Mode

Error bar mode is used to define whether the error bar line has to be drawn *horizontally*, *vertically* or in *both* side. To change the error bar mode use [errorBar.mode](#) option.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series:[{  
    //...  
    //Change the error bar mode  
    errorBar: {  
      type: "fixedValue",  
      mode: "vertical"  
    }  
  }]  
  // ...  
});
```

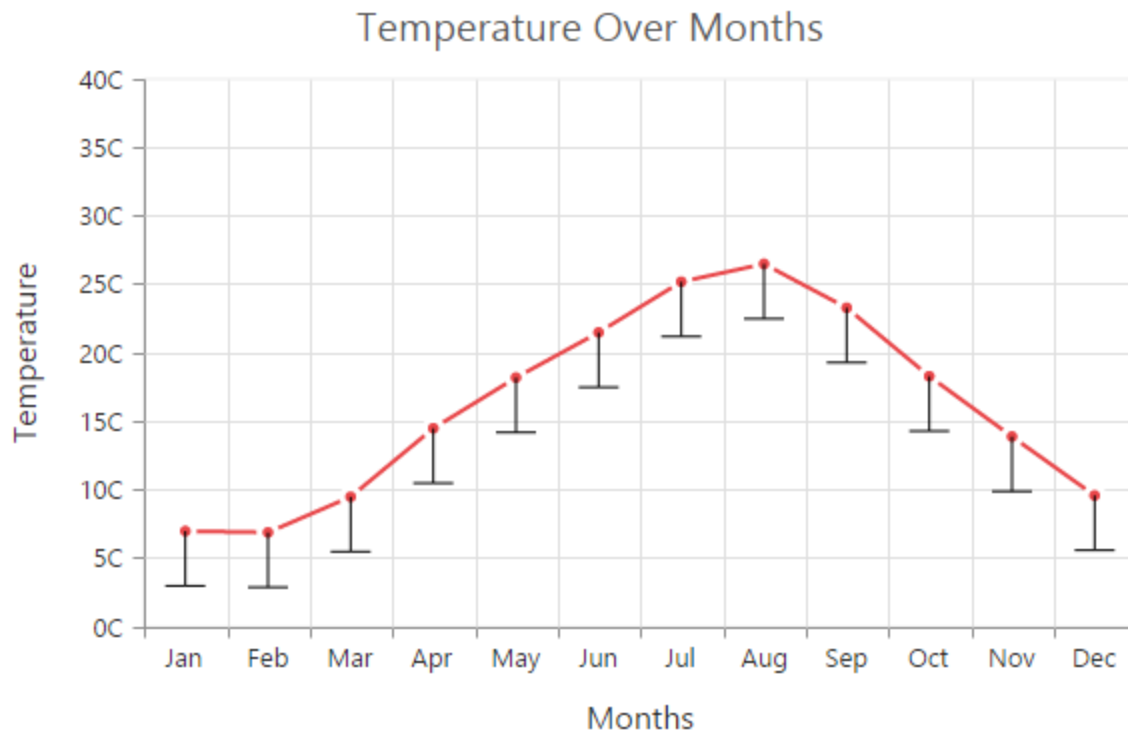


#### Changing Error Bar Direction

You can change the error bar direction to plus, minus or both side using [errorBar.directions](#) option.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series:[{  
    //...  
    //Change the error bar direction  
    errorBar: {  
      type: "fixedValue",  
      mode: "vertical",  
      direction: "minus"  
    }  
  }]  
  // ...  
});
```

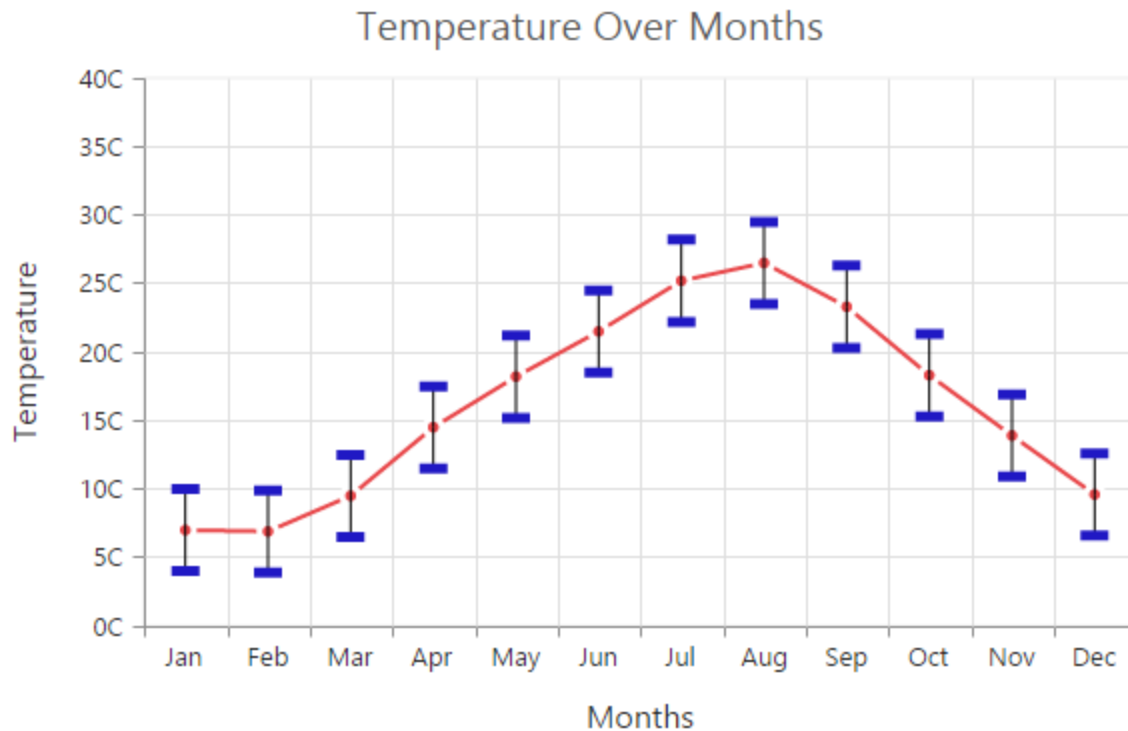


#### Customizing Error bar cap

To customize the error bar cap *visibility*, *length*, *width* and *fill* color, you can use [cap](#) option in the **series.errorBar**.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    //...
    errorBar: {
      //To customize the error bar cap
      cap:{
        visible: true,
        length: 20,
        width: 1,
        fill : "#000000"
      }
    }
  }
  // ...
});
```



#### Box and Whisker Chart

To render a Box and Whisker Chart, set the series [type](#) as “**boxAndWhisker**”.

Box and Whisker chart requires 2 fields (x and y) to plot a segment.

The field y requires n number of data or it should contains minimum of five values to plot a segment.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  {
    //...
    series:[{
      //...
      points:[ { x: "Development", y:
        [22,22,23,25,25,25,26,27,27,28,28,29,30,32,34,32,34,36,35,38]},
        { x: "Testing", y: [22,33,23,25,26,28,29,30,34,33,32,31,50]},
        { x: "HR", y: [22,24,25,30,32,34,36,38,39,41,35,36,40,56]},
        { x: "Finance", y: [26,27,28,30,32,34,35,37,35,37,45]},
        { x: "R&D", y: [26,27,29,32,34,35,36,37,38,39,41,43,58] }
      ],
      type: 'boxAndWhisker',
    }]
    //...
  });
```



### BoxPlotMode

You can change the rendering mode of the Box and Whisker series using the [boxPlotMode](#) property. The default [boxPlotMode](#) is “**exclusive**”. The other boxPlotModes available are [inclusive](#) and [normal](#).

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  //...
  series:[{
    //...
    boxPlotMode : 'inclusive',
  }]
  //...
});
```

### ShowMedian

Box and Whisker [showMedian](#) property is used to show the box and whisker average value. The default value of [showMedian](#) is “**false**”.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  //...
  series:[{
    //...
    showMedian : true,
  }]
  //...
});
```

```
//...
showMedian : true,
}]
//...
});
```



#### Customize the Outlier

Outlier symbol, width and height can be customized using outlierSettings through [outlierSettings](#) property. By default Outlier symbol is displayed as circle with a height and width of 6 pixels.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
//...
series:[{
//...
outlierSettings:
shape: 'triangle',
size:
{
width:10,
height:10
}
}]
});
```



```

}]
//...
});

```



### Pie Of Pie Chart

To render the pie of pie chart, set the series [type](#) as **pieofpie**. Pie of pie chart is used for displaying the data of a pie slice as another pie chart. The values in the second pie is displayed based on the **splitMode** property.

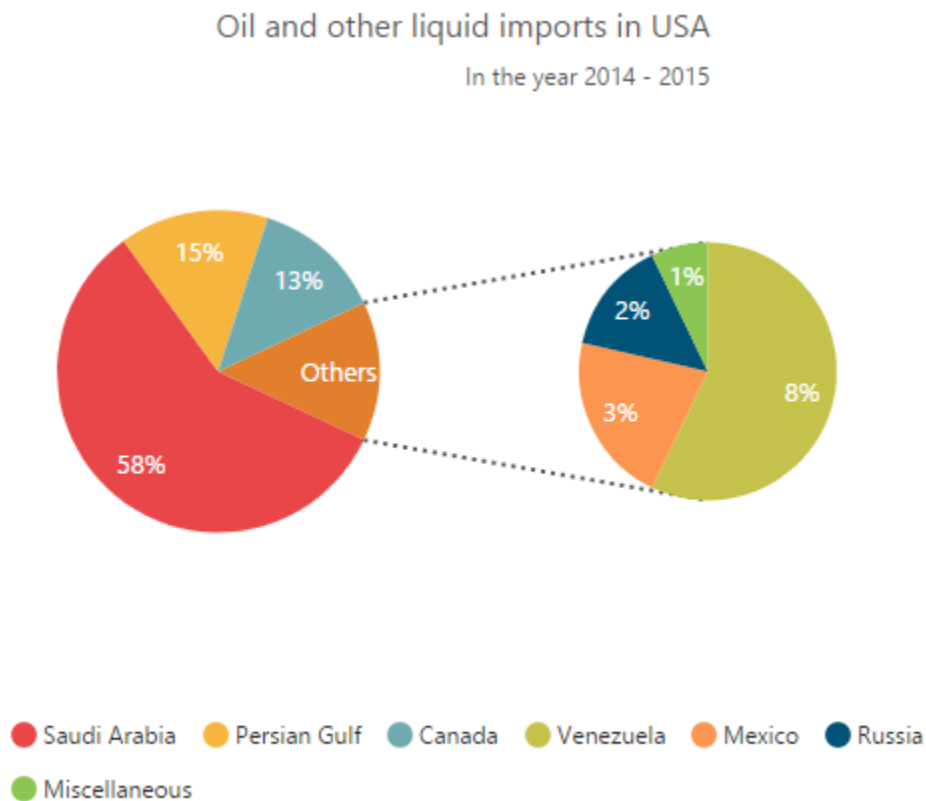
### JAVASCRIPT

```

var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  //...
  series: [{
    points: [
      {x: 'Saudi Arabia', y: 58, text: '58%'},
      {x: 'Persian Gulf', y: 15, text: '15%'},
      {x: 'Canada', y: 13, text: '13%'},
      {x: 'Venezuela', y: 8, text: '8%'},
      {x: 'Mexico', y: 3, text: '3%'},
      {x: 'Russia', y: 2, text: '2%'},
      {x: 'Miscellaneous', y: 1, text: '1%'}
    ],
    type: 'pieofpie',

```

```
splitValue:"10"
]    //..
});
```



#### Split Mode and Split Value

The points to be displayed in the second pie is decided based on the [splitMode](#) property. **SplitMode** property takes the following values.

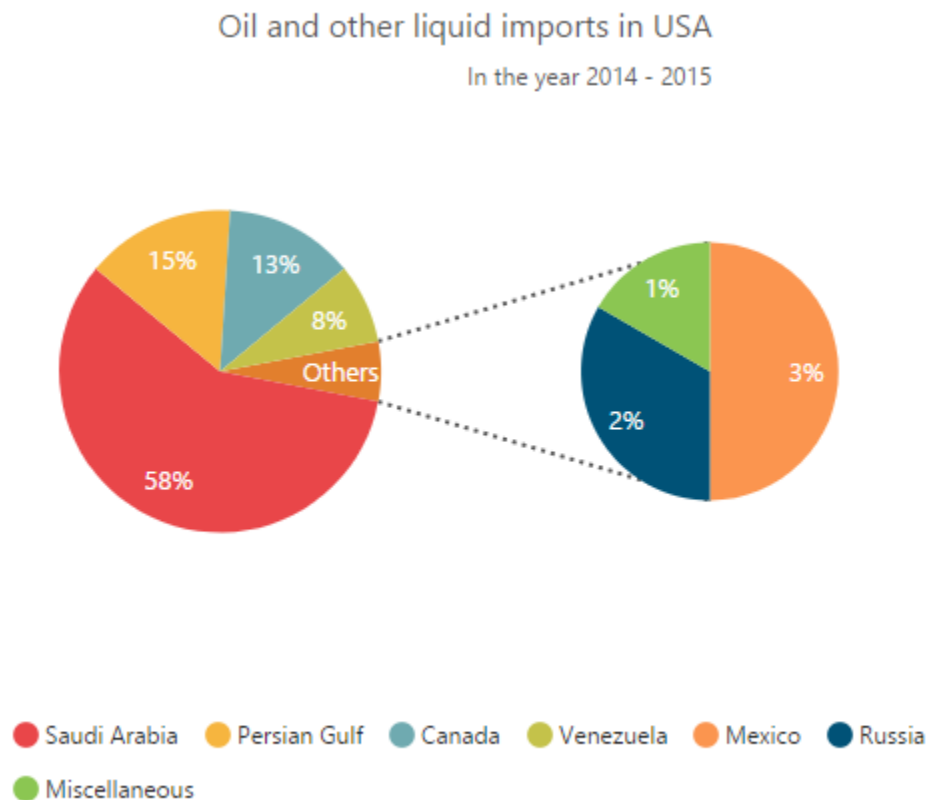
- Position – Have to split the data points based on its position
- Value – Have to split the data points based on its Y value
- Percentage – Have to split the points based on the percentage value
- Indexes – The data points with the specified indexes are split separately

By default, the splitMode is set to **Value**.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
    //..
    series: [{
        // ..
        splitMode:"Position",
        splitValue:"3"
    }]
    //..
});
```

```
});
```



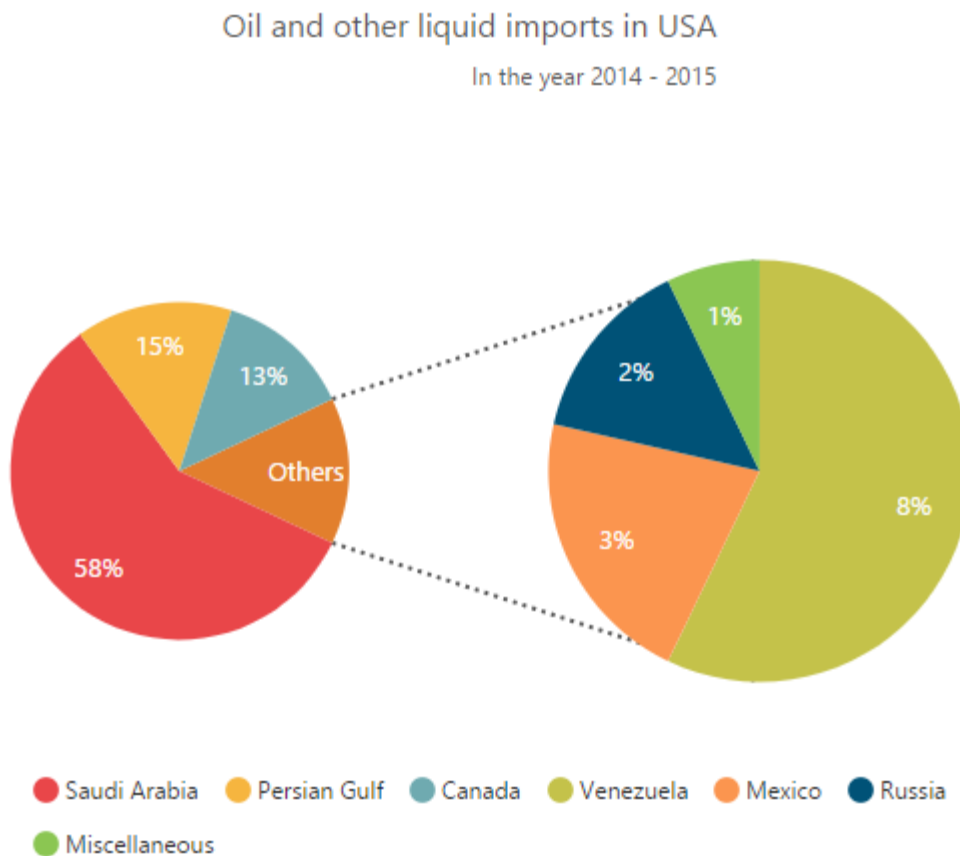
### Changing Pie Of Pie Size

The size of the second Pie can be customized by using the [pieOfPieCoefficient](#) property. The default value of `pieOfPieCoefficient` is **0.6**. Its value ranges from 0 to 1.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  //..
  series: [{
    // ..
    pieOfPieCoefficient : 1
  }] //..
});
```

The following screenshot represents the pie of pie series with `pieOfPieCoefficient` as 1

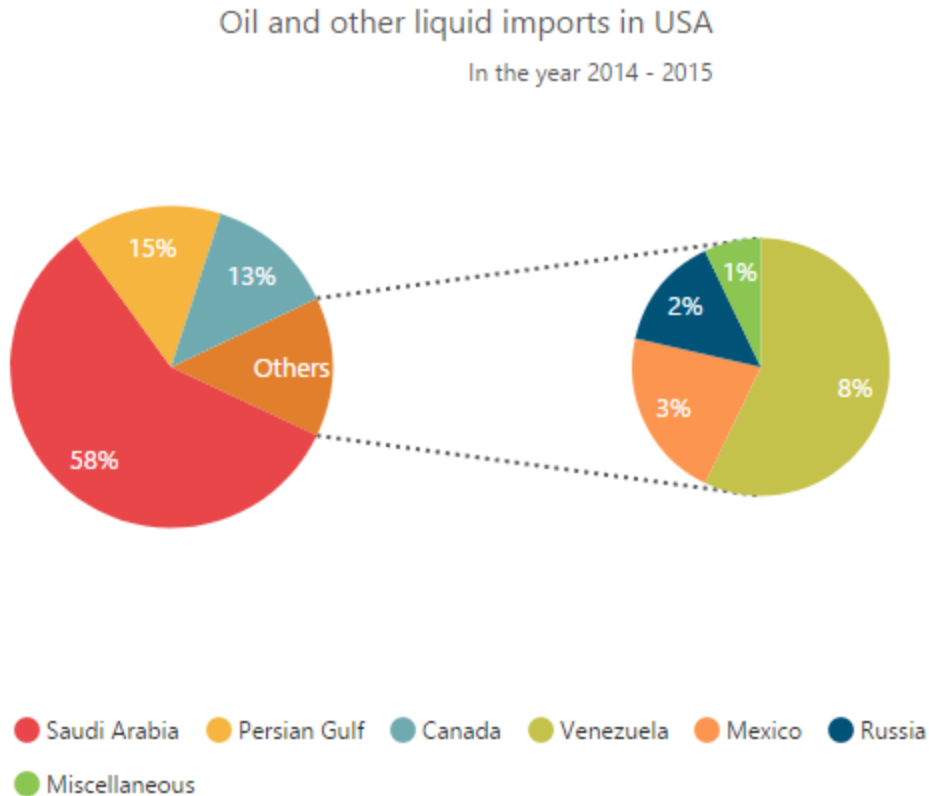


### Customizing the Gap

The distance between the two pies in the pie of pie chart can be controlled by using the [gapWidth](#) property. The default value is **50**.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  //..  
  series: [{  
    // ..  
    gapWidth:150  
  }] //..  
});
```



## Chart Series

### Multiple Series

In EjChart, you can add multiple series object in the [series](#) options. The series are rendered in the order it is added to the [series](#) option, by default. You can change this order by using the [zOrder](#) option.

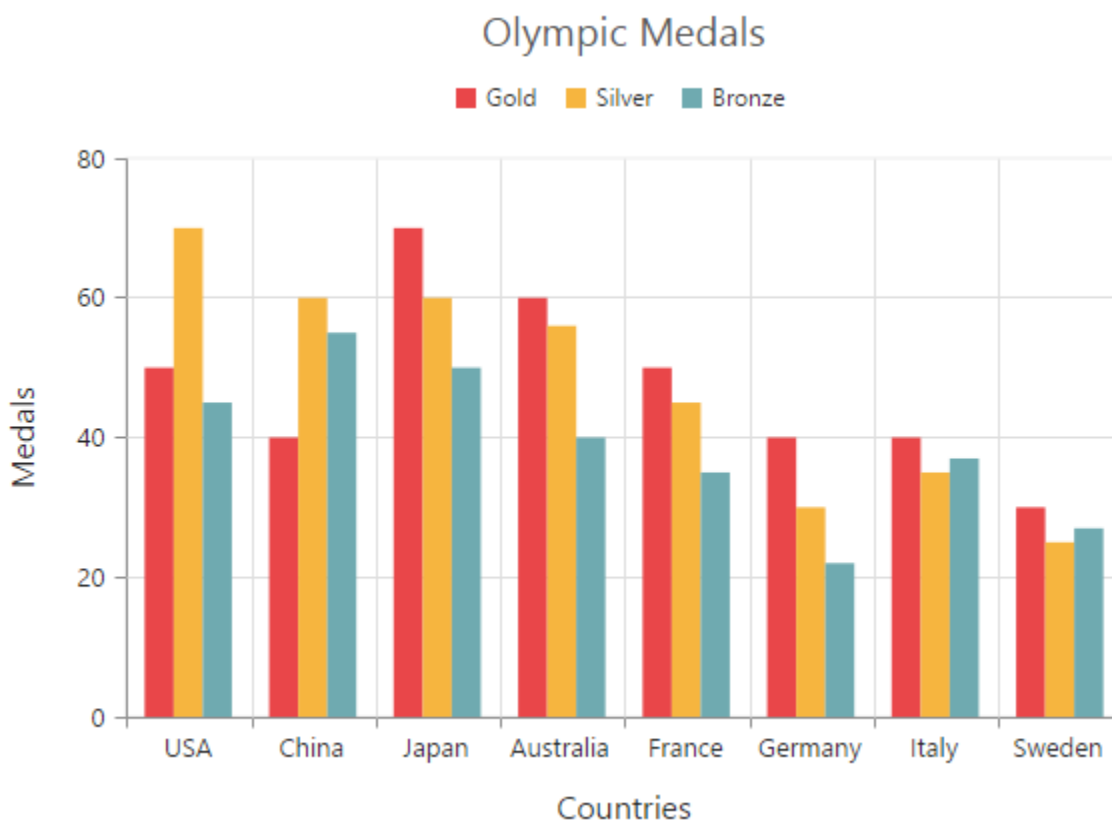
### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
  $(function () {
    var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
      // ...
      //Adding Multiple Series
      series: [{ // Add first series
        points: [{ x: "USA", y: 50 }],
        // ...
      },
      type: 'column',
      // ...
    ],
    { // Add second series
      points: [{ x: "USA", y: 70 }],
      // ...
    },
    type: 'column'
  )
}

```

```
// ...
},
{
    // Add third series
    points: [{ x: "USA", y: 45 },
    // ...
    ],
    type: 'column'
    // ...
}],
// ...
});
});
}
```



#### Customizing all series together

By using the [commonSeriesOptions](#), you can customize the series options for all the series commonly, instead of setting the options directly on each series object.

---

**Note:** The inline properties of the series has the first priority and override the commonSeriesOptions.

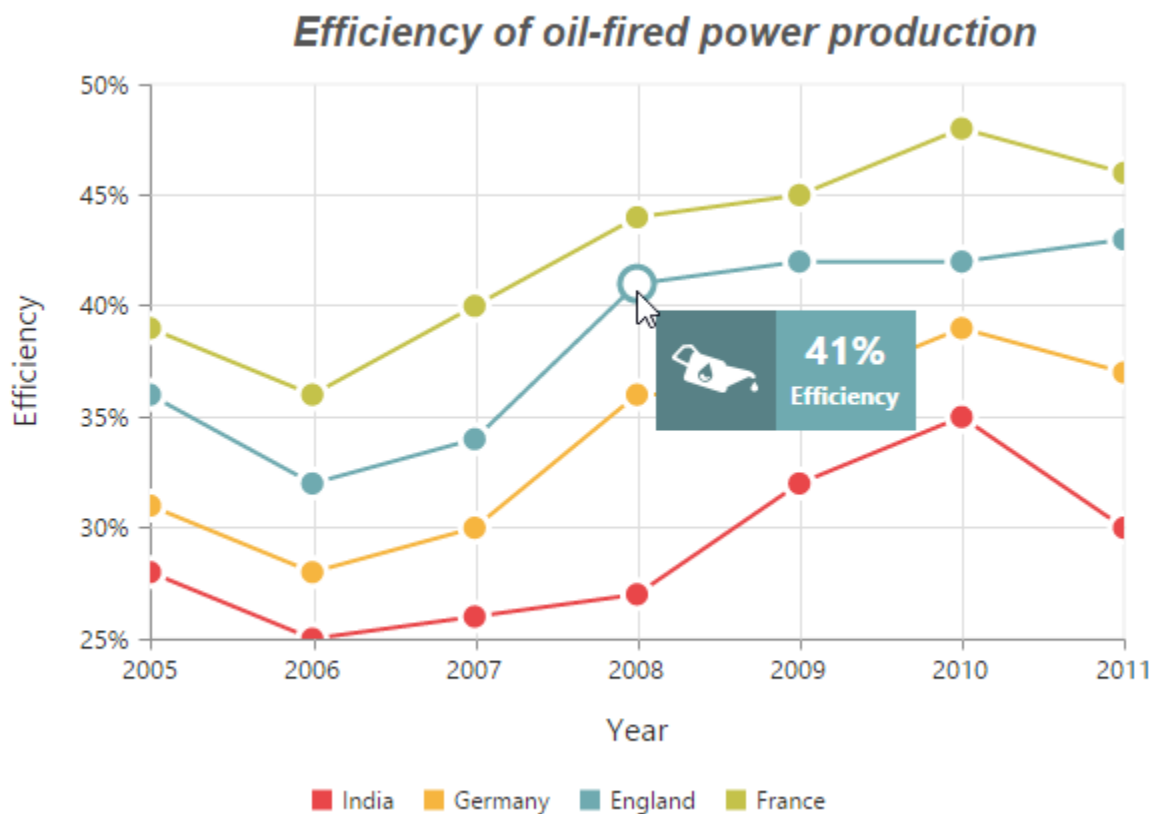
---

The following code example explains on how to enable marker, tooltip and animation for the chart series by using the commonSeriesOptions.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
    // ...
});
```

```
//Initializing Common Properties for all the series
commonSeriesOptions: {
  type: 'line',
  enableAnimation: true,
  tooltip: {
    visible: true,
    template: 'Tooltip'
  },
  marker: {
    shape: 'circle',
    size: {
      height: 10, width: 10
    },
    visible: true
  },
  border: { width: 2 }
},
series: [{
  // ...
}, {
  // ...
}],
// ...
});
```

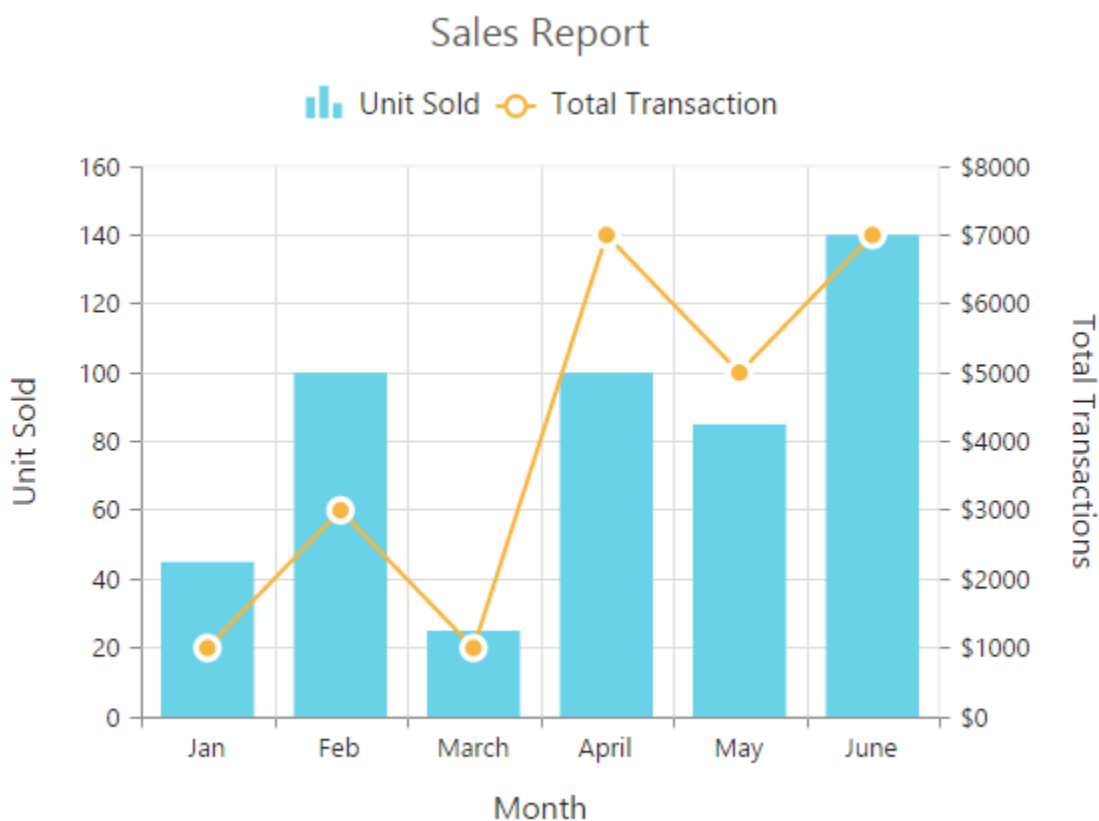


### Combination Series

EjChart allows you to render the combination of different series in the chart.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  series: [{
    //Set chart type to series1
    type: 'column',
    // ...
  }, {
    //Set chart type to series2
    type: 'line',
    // ...
  }],
  // ...
});
```



#### Limitation of combination chart

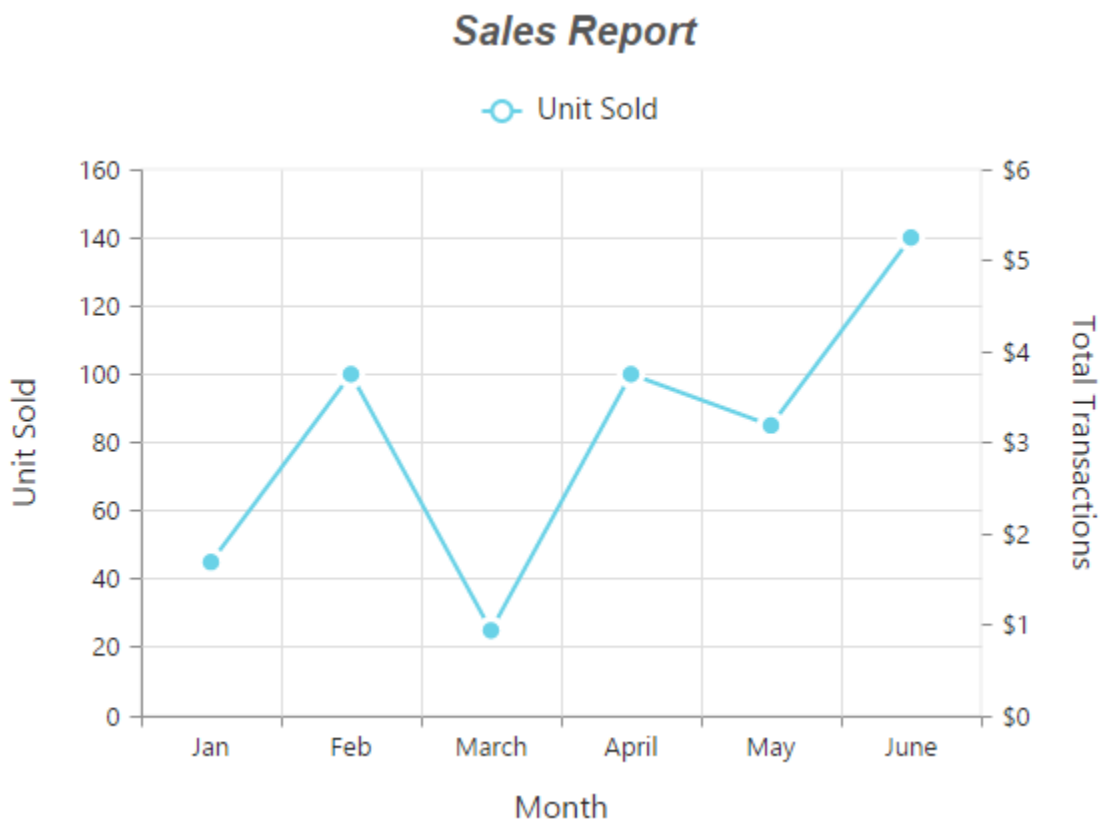
- [Bar](#), [StackingBar](#), and [StackingBar100](#) cannot be combined with the other Cartesian type series.
- Cartesian type series cannot be combined with the accumulation series ([pie](#), [doughnut](#), [funnel](#), and [pyramid](#)).
- [Polar](#) and [Radar](#) series cannot be combined with the accumulation and Cartesian type series.



When the combination of Cartesian and accumulation series types are added to the series option, the series that are similar to the first series are rendered and other series are ignored. The following code example illustrates this,

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  //Adding Multiple Series
  series: [{ // Add line series
    points: [{ x: "Jan", y: 45 },
    // ...
  ],
  type: 'line',
  // ...
},
{ // Add [Pie](chart-types#pie-chart) series
  points: [{ x: "Jan", y: 70 },
  // ...
  ],
  type: 'pie'
  // ...
}],
  // ...
});
```



## Data Markers

Data markers are used to provide information about the data point to the user. You can add a shape and label to adorn each data point.

### Add Shapes

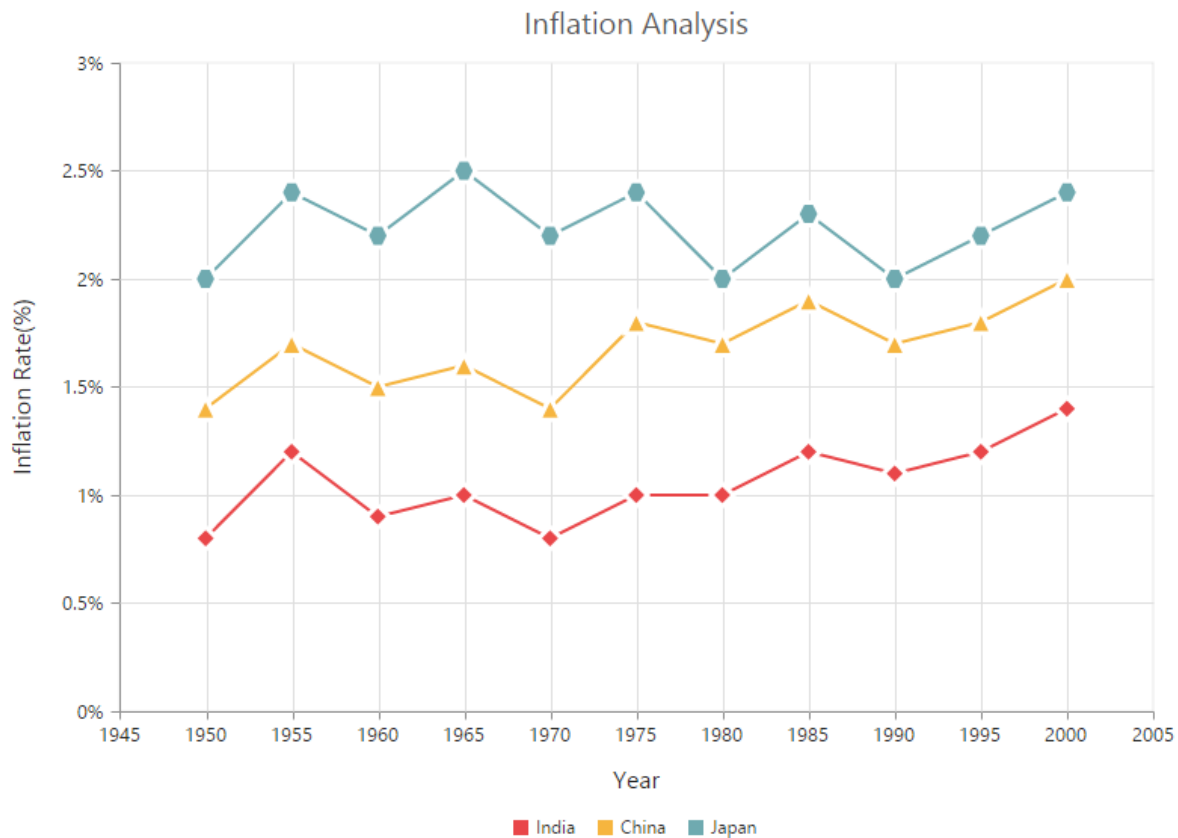
You can add shapes to any chart types but they are often used with line, area and spline series to indicate each data point. It is highlighted when you hover the mouse on the shape.

Shapes can be added to the chart by enabling the [visible](#) option of the [marker](#) property. There are different shapes you can add to the chart by using the shape option such as rectangle, circle, diamond etc.

The following code example explains on how to enable series marker and add shapes,

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
  $(function () {
    var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
      // ...
      //Adding shapes to series
      series: [{
        // ....
        marker: {
          shape: 'Diamond',
          visible: true
        }
      },
      {
        // ...
        marker: {
          shape: 'Triangle',
          visible: true
        }
      },
      {
        // ...
        marker: {
          shape: 'Hexagon',
          visible: true
        }
      }
    ]],
    // ....
  });
});
}
```



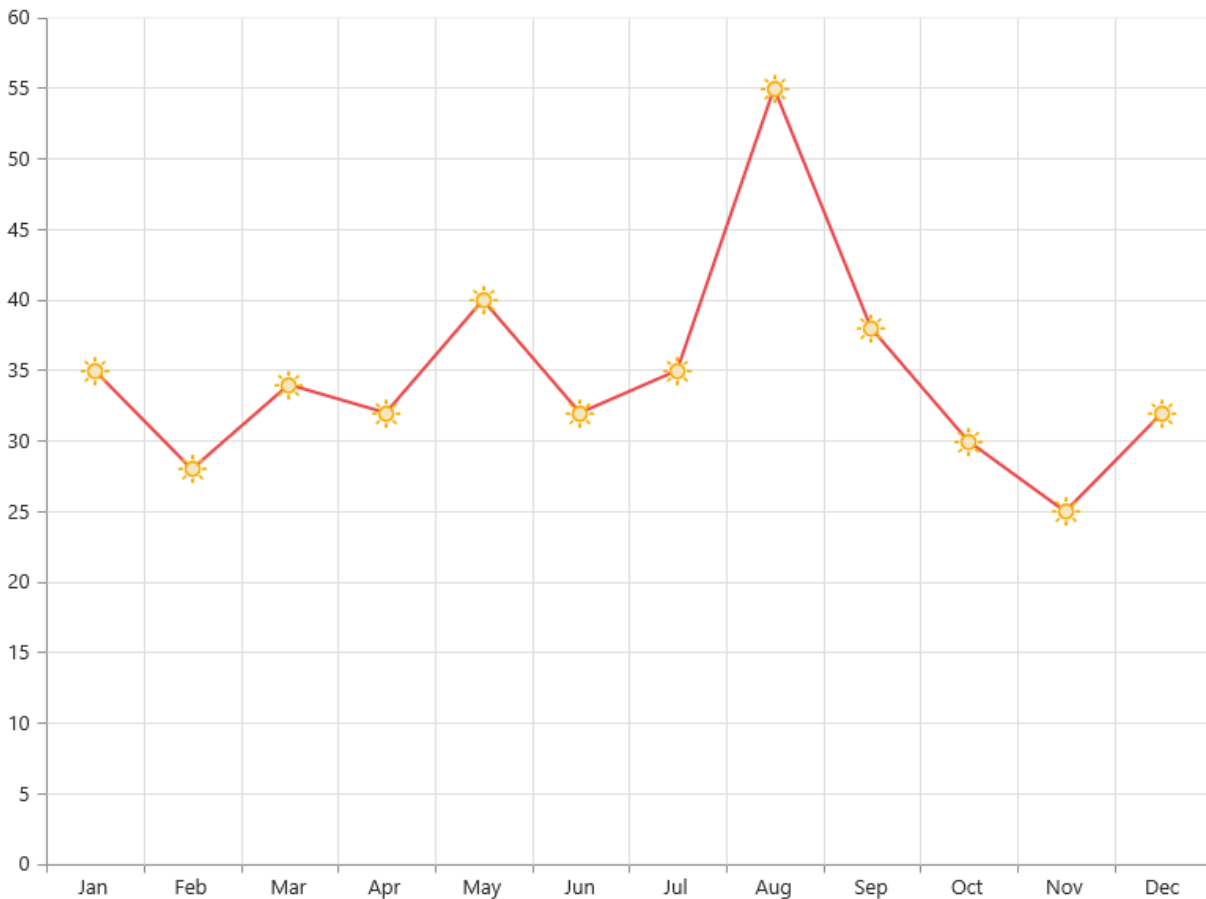
Add image as marker

Apart from the shapes, you can also add images to mark the data point by using the [imageUrl](#) option.

The following code example illustrates this,

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // ...
    marker: {
      // Enable and customize the marker shape and size
      visible: true,
      // In order to set imageUrl, set shape as 'image' .
      shape: "image",
      imageUrl: "sun_annotation.png",
      size: {width: 20, height: 20}
    }
  }],
  // ...
});
```



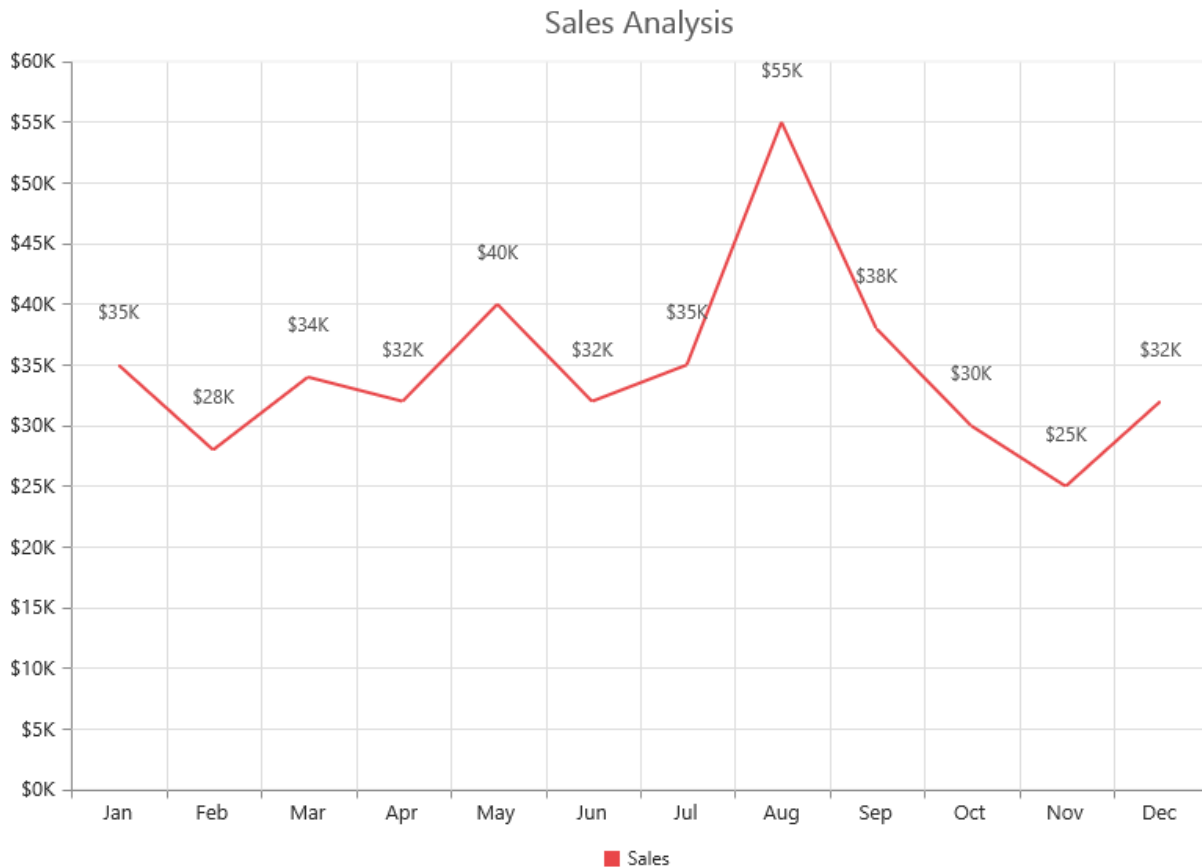
### Add labels

Data label can be added to a chart series by enabling the [visible](#) property in the [dataLabel](#) option. The labels appear at the top of the data point, by default.

The following code example shows how to enable data label and set its horizontal and vertical text alignment.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // ...
    marker: {
      dataLabel: {
        //Set text alignment to data label text
        visible: true,
        horizontalTextAlignment: "center",
        verticalTextAlignment: "far"
      }
    }
  }],
  // ...
});
```



Label content can be formatted by using the template option. Inside the template, you can add the placeholder text “*point.x*” and “*point.y*” to display corresponding data points x & y value.

You can adorn the labels with background shapes by setting *shape* option.

The following code example shows how to add background shapes and set template to data label.

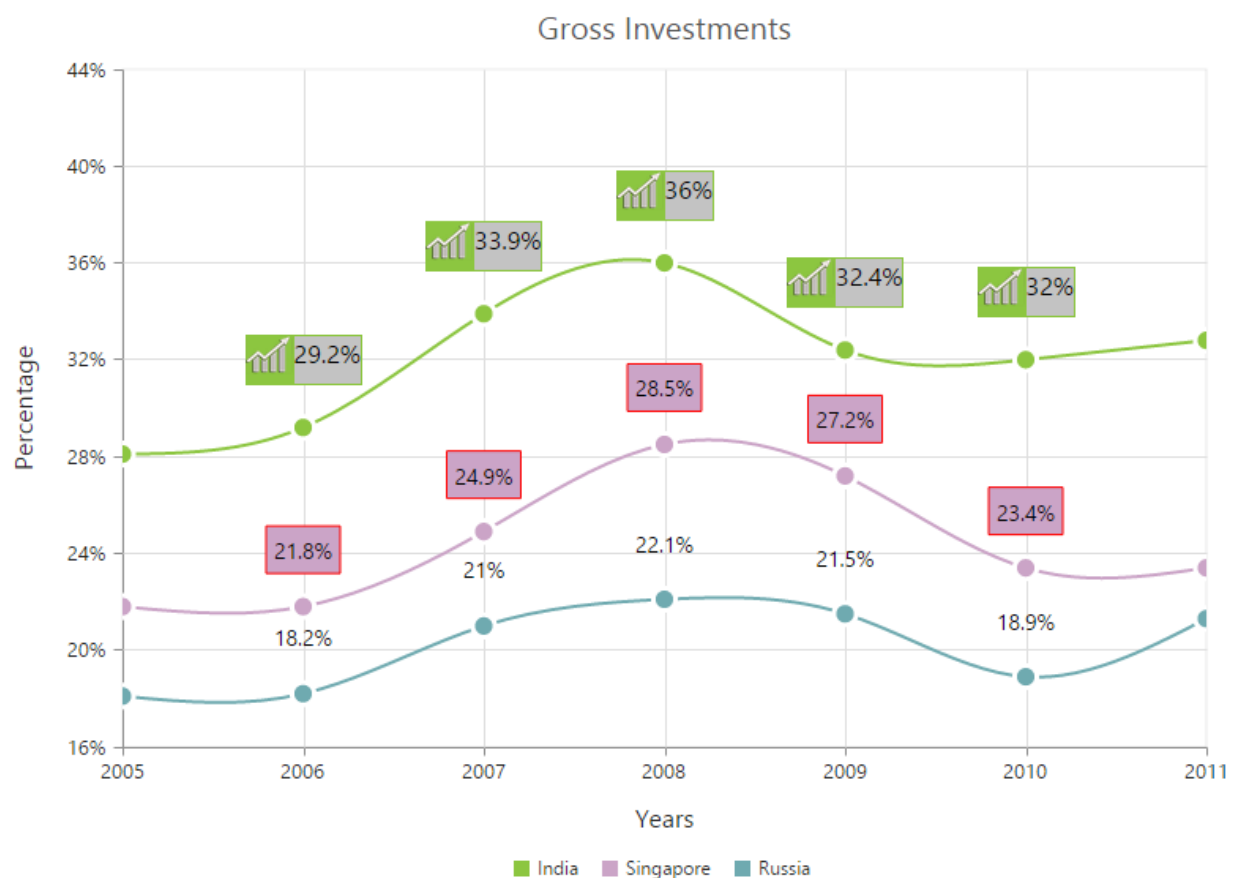
### HTML

```
<div id="template">
  <div id="left">
    
  </div>
  <div id="right">
    <div id="point">#point.y#</div>
  </div>
</div>
```

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // ...
    marker: {
      dataLabel: {
        visible: true,
```

```
//Set template to data label
template: 'template'
}}
}, {
// ...
marker: {
dataLabel: {
visible: true,
//Add background shape to the data label
shape: 'Rectangle',
border: { width: 1, color: "red" }
}}
}, {
// ...
marker: {
dataLabel: {
visible: true
}}
}},
// ...
});
```



The appearance of the labels can be customized by using the [font](#) and [offset](#) options. The [offset](#) option is used to move the labels vertically. Also, labels can be rotated by using the [rotate](#) option.

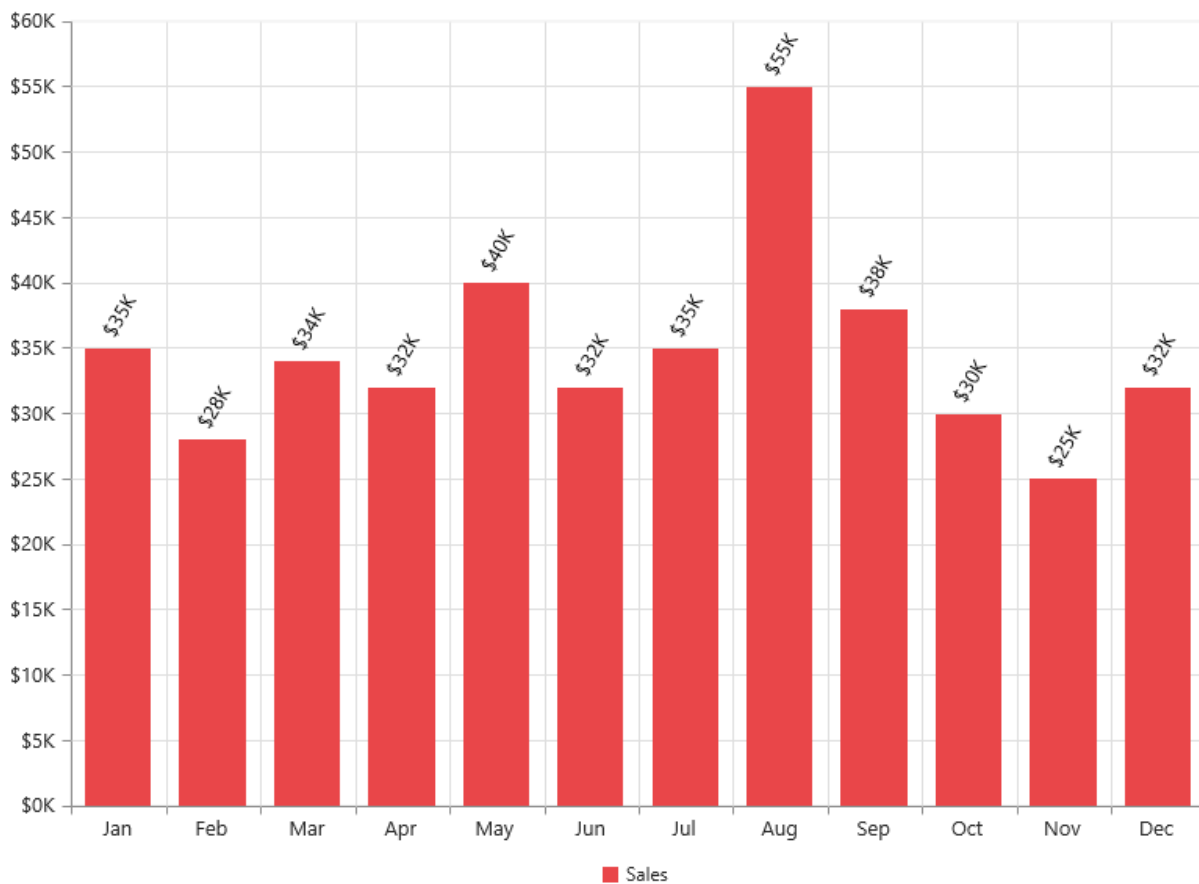
The following code example shows how to rotate data label text and customize the font.

**JAVASCRIPT**

```

var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // ...
    marker: {
      dataLabel: {
        visible: true,
        //Rotate data label and customize the font
        angle: "300",
        offset: 15,
        font: { color: "black", size:"13px" }
      }
    }
  }],
  // ...
});

```



You can position the label to the top, center or bottom position of the segment by using the [textPosition](#) option for the chart types such as column, bar, stacked bar, stacked column, 100% stacked bar, 100% stacked column, candle and OHLC.

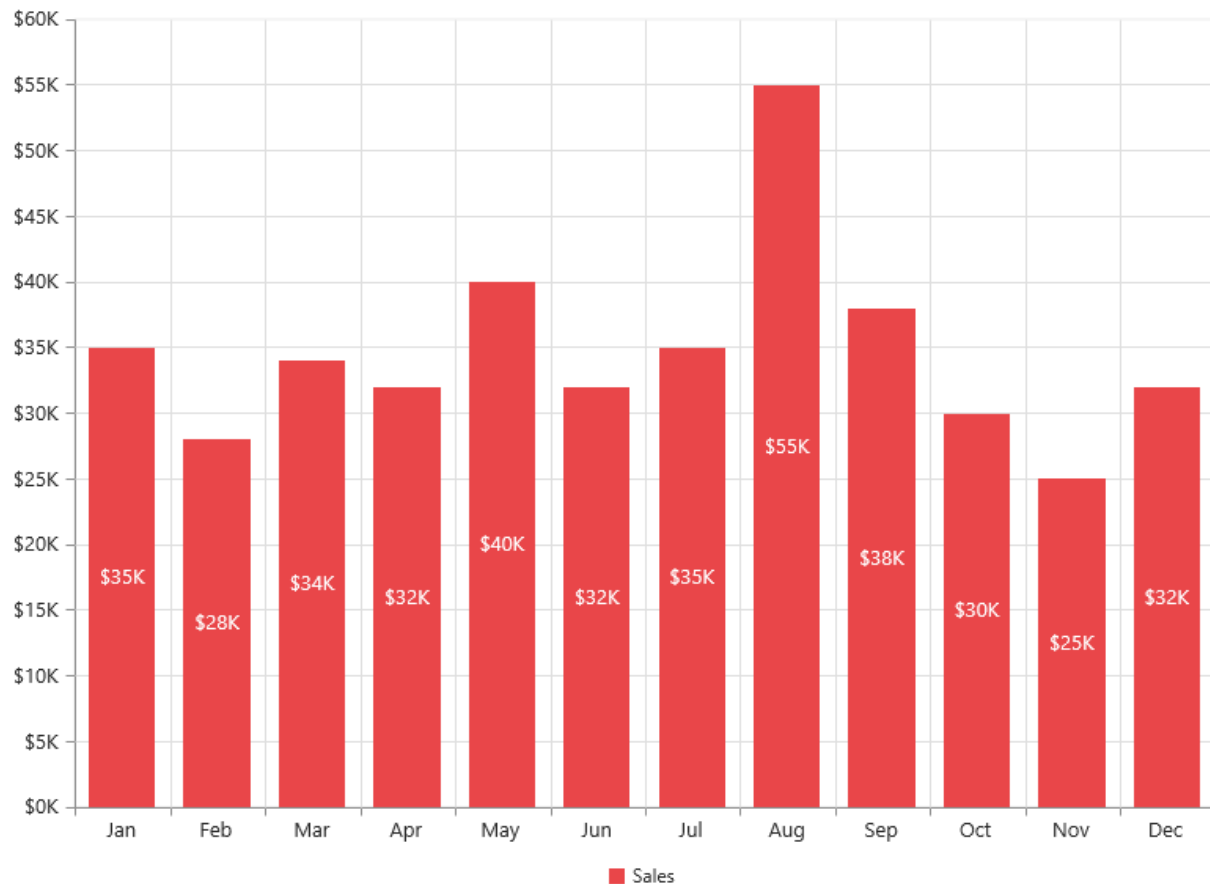
The following code example shows how to set `textPosition` to display data label in the middle of the column rectangle.

**JAVASCRIPT**

```

var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    // ....
    marker: {
      dataLabel: {
        visible: true,
        // Place the data label text position in the centre of the rectangle
        textPosition: "middle"
        // ...
      }
    }
  }],
  // .....
});

```



The label can be positioned inside or outside the perimeter of the series by using the [labelPosition](#) option for the chart types such as Pie and Doughnut, .

The following code example shows how to set the *labelPosition*,

**JAVASCRIPT**

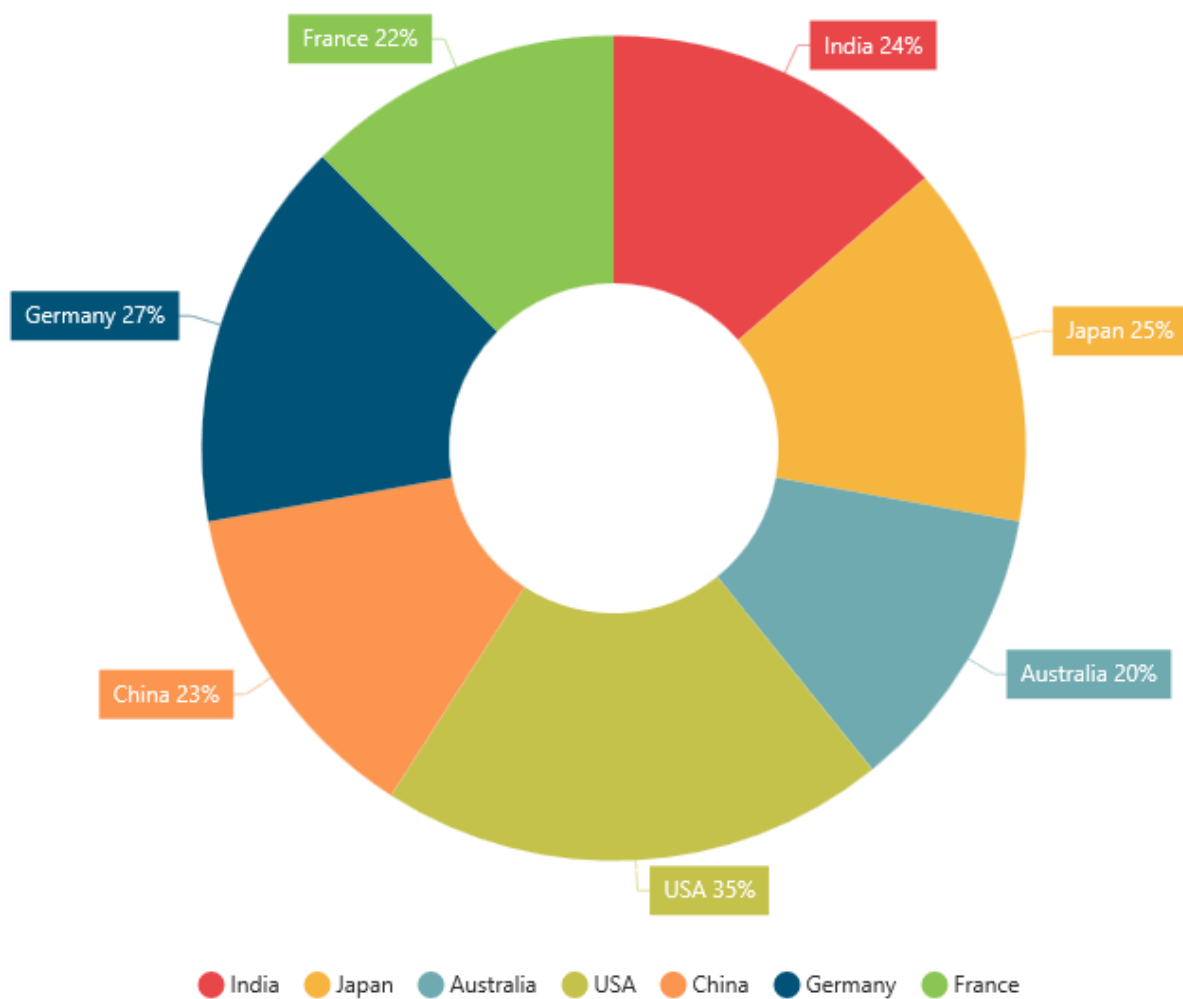
```

var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...

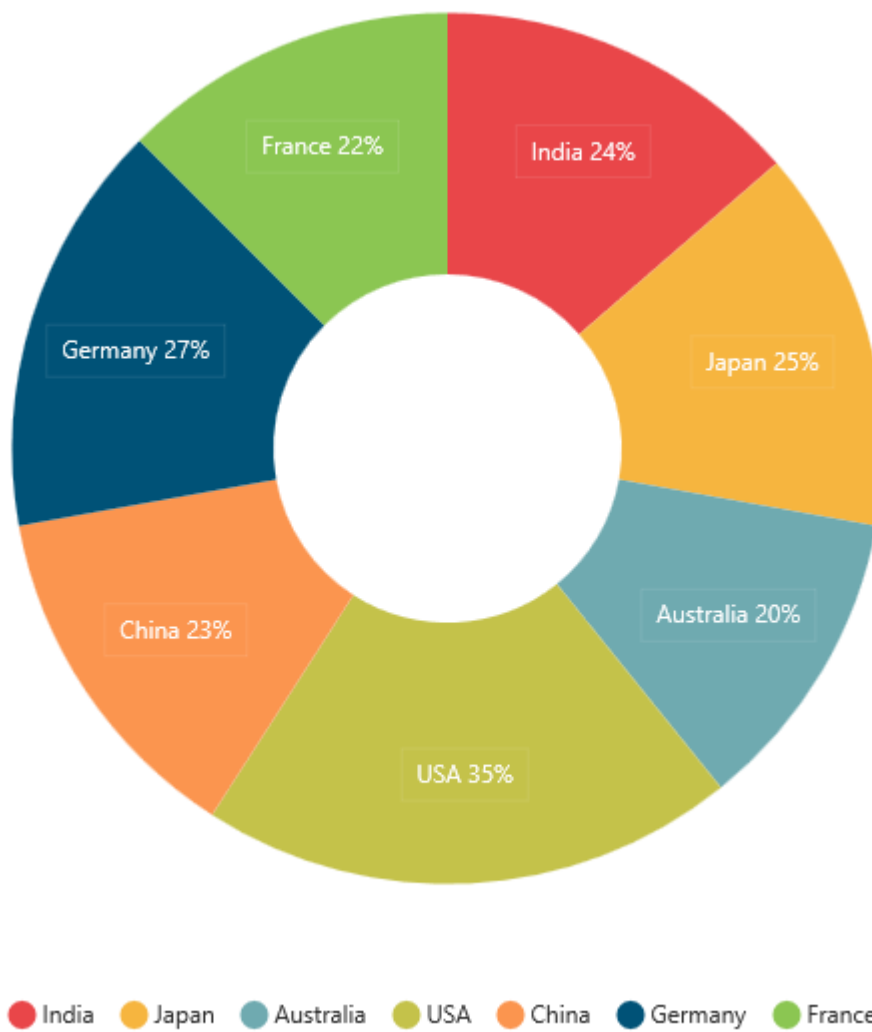
```



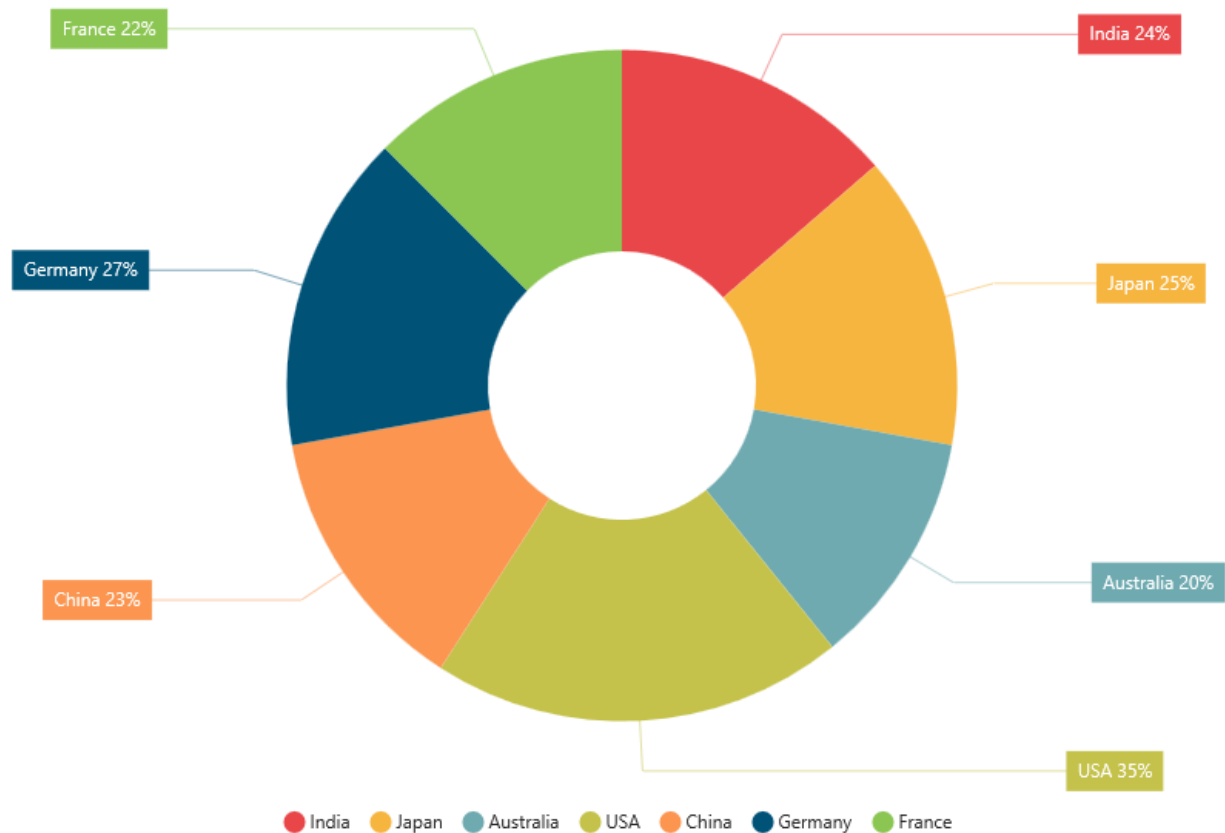
```
series:[{
  points: [{ x: 'India', y: 24, text: 'India 24%' },
    { x: 'Japan', y: 25, text: 'Japan 25%' },
    { x: 'Australia', y: 20, text: 'Australia 20%' },
    { x: 'USA', y: 35, text: 'USA 35%' },
    { x: 'China', y: 23, text: 'China 23%' },
    { x: 'Germany', y: 27, text: 'Germany 27%' },
    { x: 'France', y: 22, text: 'France 22%' }],
  marker: {
    dataLabel: {
      visible: true,
      shape: 'rectangle',
      font: {color: "white"}
    }
  },
  type: 'doughnut',
  //Display data label outside position
  labelPosition: 'outside'
}],
// ...
});
```



The following screenshot displays the labels when the [labelPosition](#) is set as *inside* position.



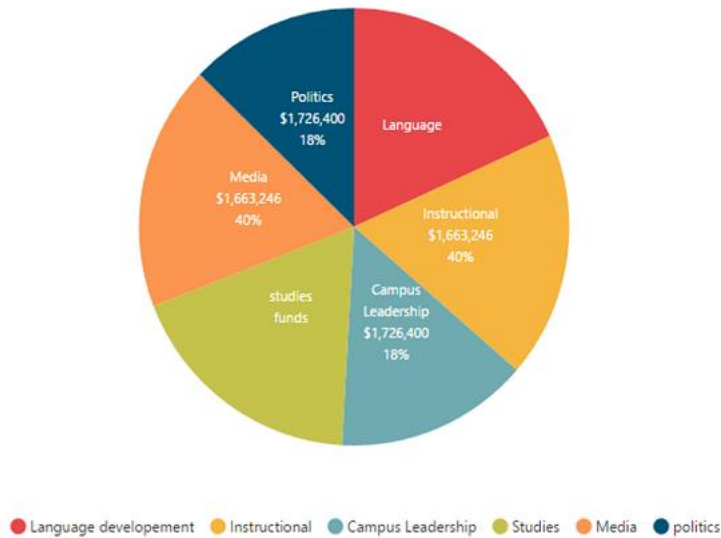
The following screenshot displays the labels when the [labelPosition](#) is set as *outsideExtended* position.



The label can be wrapped for pie, doughnut, funnel, and pyramid series by setting the `enableWrap` property.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // . . .
  series:[
    {
      // . . .
      marker: {
        dataLabel: {
          // enable the dataLabel
          visible: true,
          // enable the wrapping option
          enableWrap: true,
          // set the maximumLabelWidth of the data label
          maximumLabelWidth: 32
        }
        // . . .
      }
    }
  ]
  // . . .
});
```



### Customize specific points

By using the ejChart, you can also customize the individual/specific markers with different colors, shapes and also with different images.

There are two ways to achieve this based on how the data is fed to the series.

When the data is provided by using the [points](#) option, you can add marker for each data point or specific point by using the [marker](#) option as illustrated in the following code example.

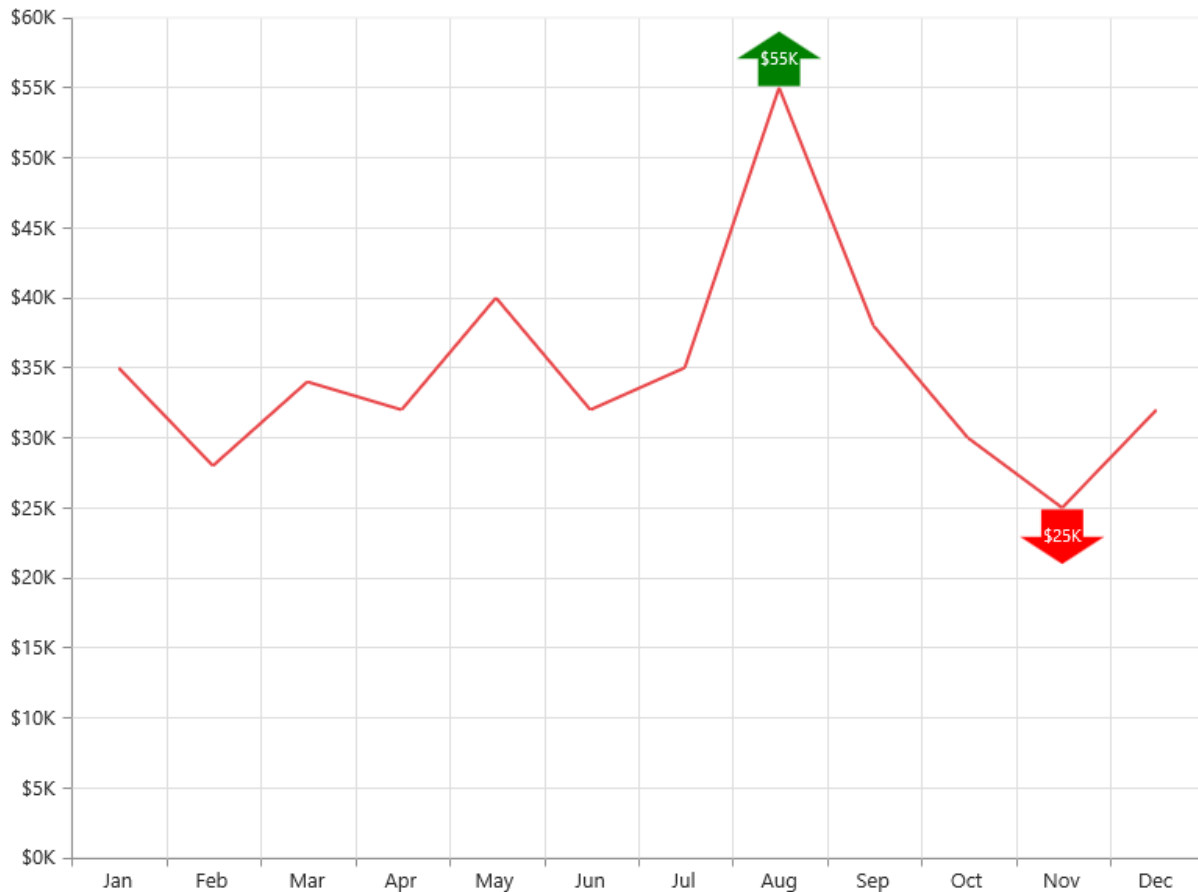
### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    // ...
    points: [ { x: 'Jan', y: 35 }, { x: 'Feb', y: 28 }, { x: 'Mar', y: 34 },
              { x: 'Apr', y: 32 }, { x: 'May', y: 40 }, { x: 'Jun', y: 32 },
              { x: 'Jul', y: 35 }, { x: 'Aug', y: 55 },
    marker: {
      //Enable and customize the data label for a point
      dataLabel: {
        visible: true,
        offset: -10,
        shape: "upArrow", font: { color: "white", size: '11px' },
        margin: { left: 15, right: 15, top: 10, bottom: 10 },
        fill: "green"
      }
    },
    { x: 'Sep', y: 38 }, { x: 'Oct', y: 30 },
    { x: 'Nov', y: 25 },
    marker: {
      //Enable and customize the data label for a point
      dataLabel: {
        visible: true,
        offset: -22,
        verticalTextAlignment: 'near',
        shape: "downArrow", font: { color: "white", size: '11px' },
        margin: { left: 15, right: 15, top: 10, bottom: 10 },
```

```

fill: "red"
} } }, { x: 'Dec', y: 32 }],
}],
// ...
});

```



When the data is bound to the series by using the [dataSource](#) option, you can customize the points in the [seriesRendering](#) event as illustrated in the following code example,

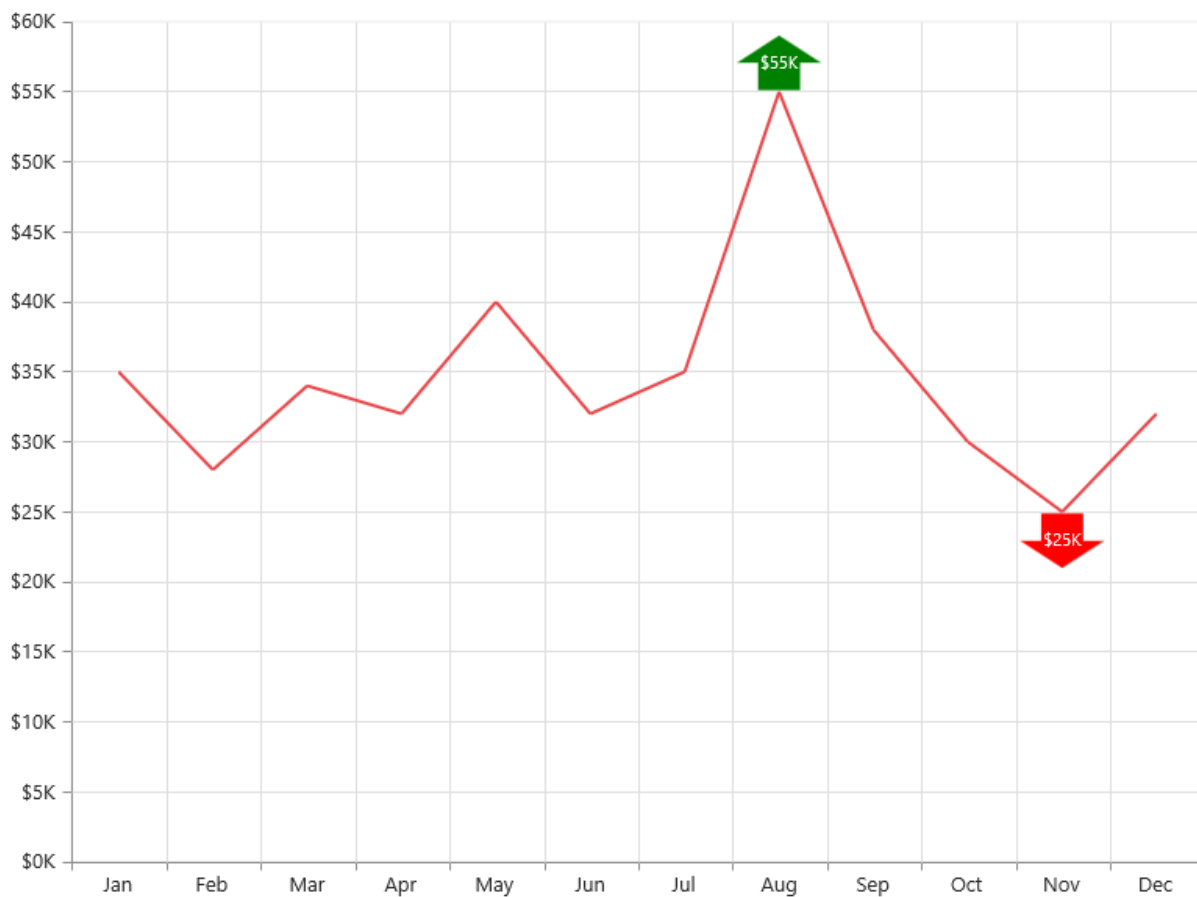
#### JAVASCRIPT

```

var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    //Add datasource and set xName and yName
    dataSource: chartData,
    xName: "month",
    yName: "sales"
  }],
  //Fires on rendering the series
  seriesRendering: "onSeriesRender",
  // ...
});
//Define the seriesRendering client side event
function onSeriesRender(sender) {

```

```
//Enable and customize the dataLabel for a point using event
sender.data.series.points[7].marker = {
  dataLabel: {
    visible: true,
    offset: -10,
    shape: "upArrow", font: { color: "white", size: '11px' },
    margin: { left: 15, right: 15, top: 10, bottom: 10 },
    fill: "green"
  }
};
sender.data.series.points[10].marker = {
  //Enable and customize the dataLabel for a point using event
  dataLabel: {
    visible: true,
    offset: -22,
    verticalTextAlignment: 'near',
    shape: "downArrow", font: { color: "white", size: '11px' },
    margin: { left: 15, right: 15, top: 10, bottom: 10 },
    fill: "red"
  }
};
}
```



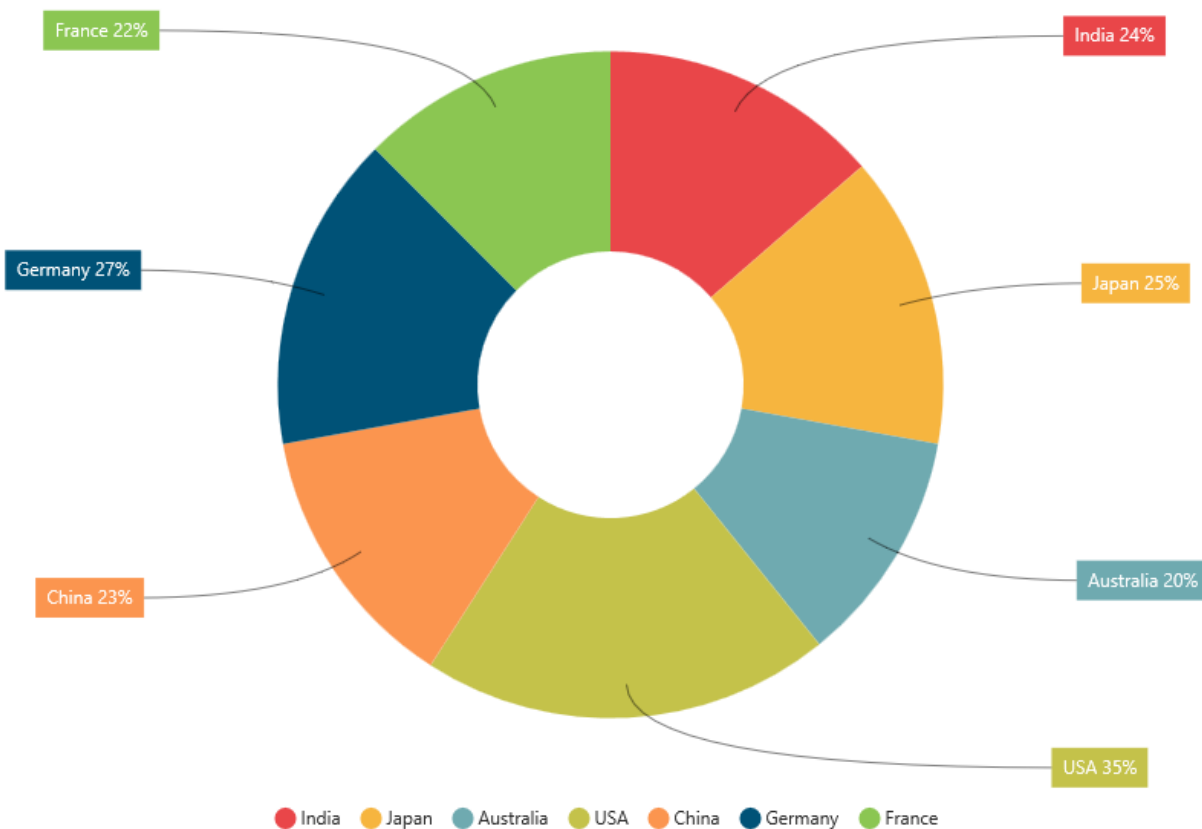
### Connect Line

This feature is used to connect label and data point by using a line. It can be enabled only for Pie, Doughnut, Pyramid and Funnel chart types. Connector line types can be set as *bezier* or *line* by using the [type](#) option.

The following code example illustrates this,

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series: [{  
    // ...  
    marker: {  
      dataLabel: {  
        visible: true,  
        // Set connector line type and customize the color,  
        connectorLine: { type: 'bezier', color: 'black' }  
        // ...  
      } },  
      // ...  
      labelPosition: 'outsideExtended',  
    } ],  
    // ...  
  });
```





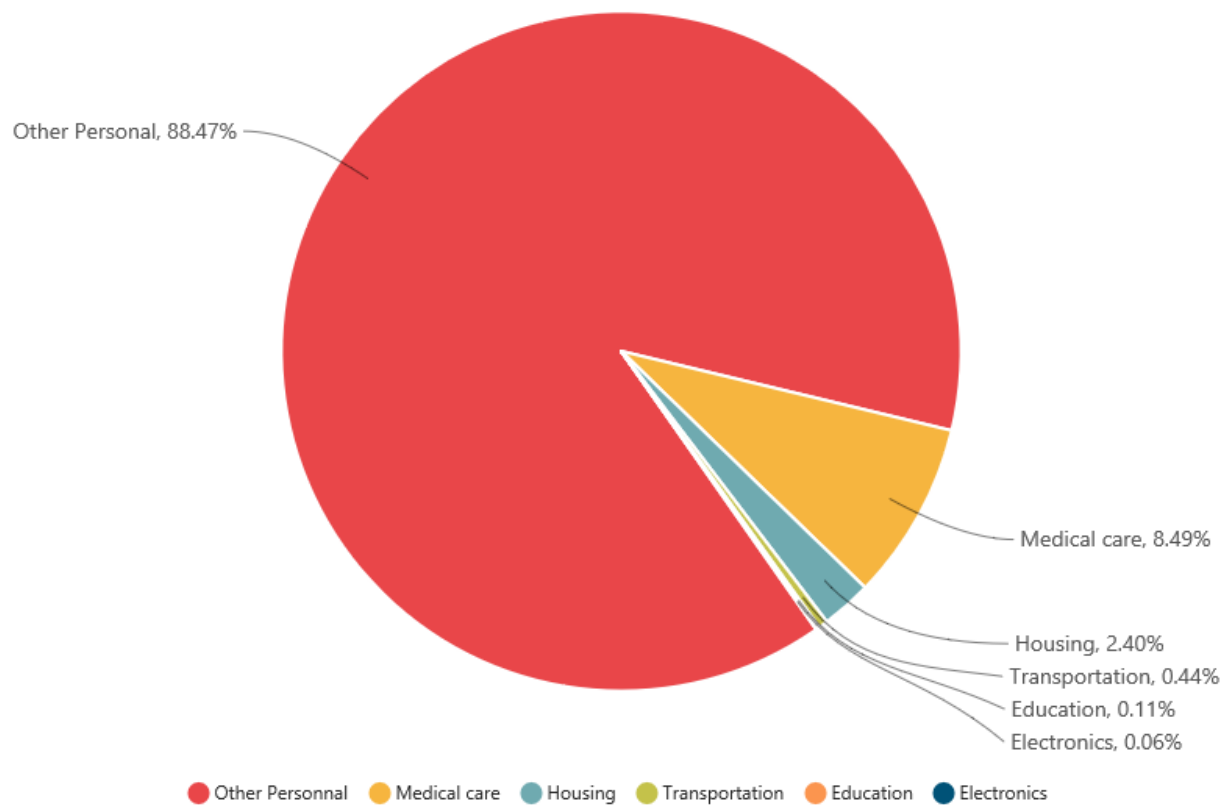
### Smart labels

Overlapping of the labels can be avoided by enabling the [enableSmartLabels](#) property. The default value is *true* for *accumulation type series* and *false* for *other series types*.

The following code example shows how to enable smart labels,

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  //Initializing Series
  series: [{
    points: [{ x: 'Other Personal', y: 94658, text: 'Other Personal, 88.47%' },
      { x: 'Medical care', y: 9090, text: 'Medical care, 8.49%' },
      { x: 'Housing', y: 2577, text: 'Housing, 2.40%' },
      { x: 'Transportation', y: 473, text: 'Transportation, 0.44%' },
      { x: 'Education', y: 120, text: 'Education, 0.11%' },
      { x: 'Electronics', y: 70, text: 'Electronics, 0.06%' }],
    marker: {
      dataLabel: {
        visible: true,
        shape: 'none',
        connectorLine: { type: 'bezier', color: 'black' },
        font: { size: '14px' }
      },
      border: { width: 2, color: 'white' },
      type: 'pie',
      labelPosition: "outsideExtended",
      //Display data label outside position and enable smart labels
      enableSmartLabels: true
    },
    // ...
  }];
```



## Legend

The legend contains the list of chart series and Trendlines that appear in a chart.

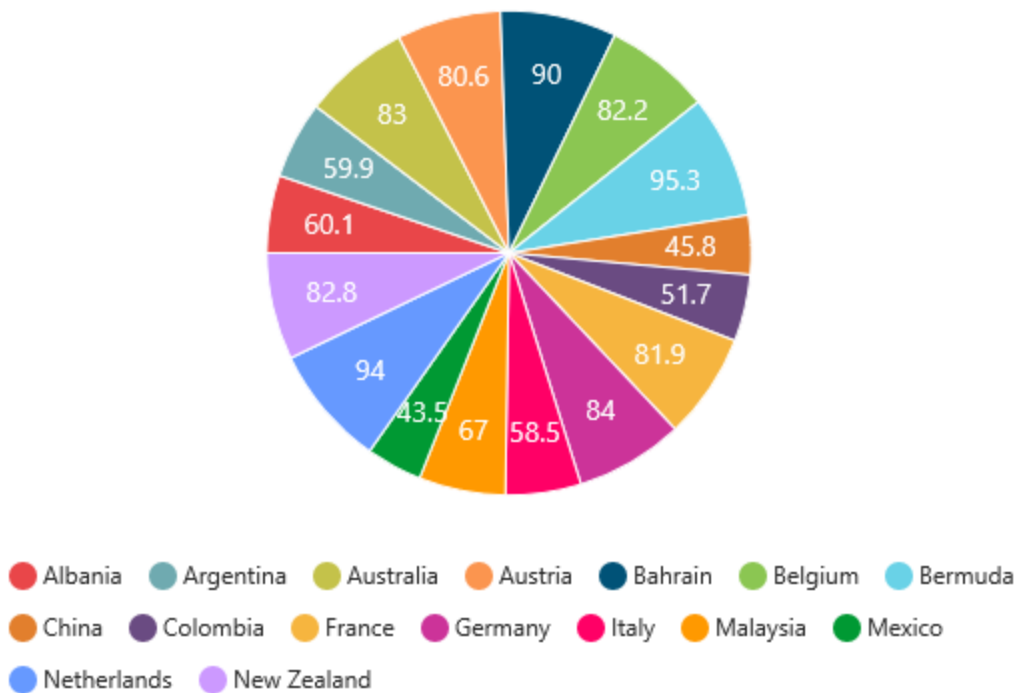
### Legend Visibility

By default, the legend is enabled in the chart. You can enable or disable it by using the [visible](#) option of the legend.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
  $(function () {
    var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
      // ...
      legend: {
        //Visible chart legend
        visible: true
      },
      //...
    });
  });
}
```

Internet users (per 100 people) in 2013



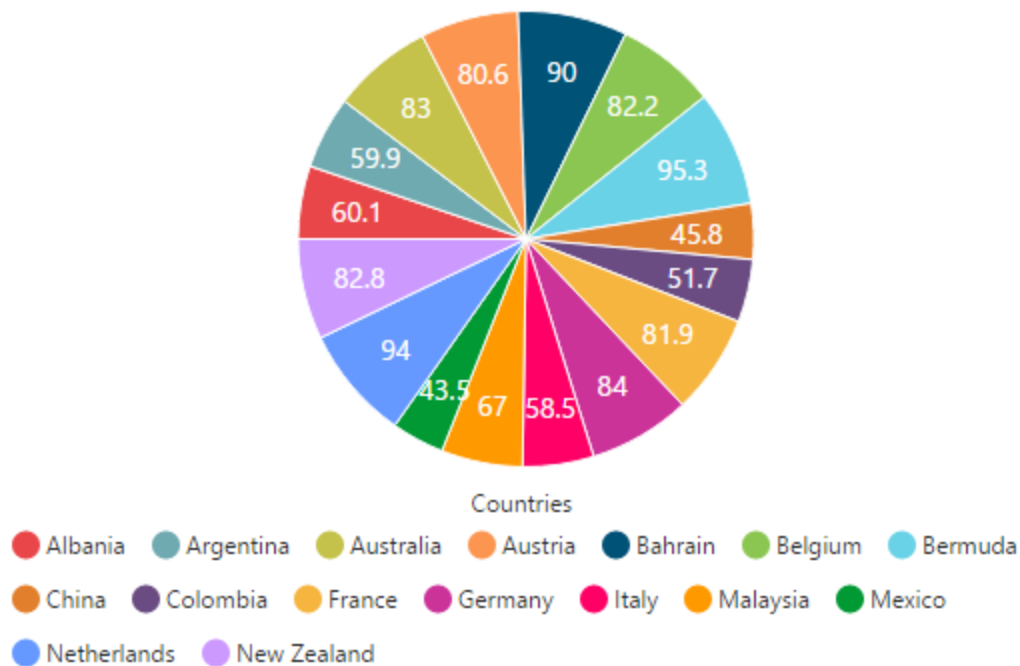
### Legend title

To add the title to the legend, you have to specify the [legend.title.text](#) option.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  legend: {
    //...
    title: {
      //Add title to the chart legend
      text: "Countries",
    },
  },
  // ...
});
```

Internet users (per 100 people) in 2013



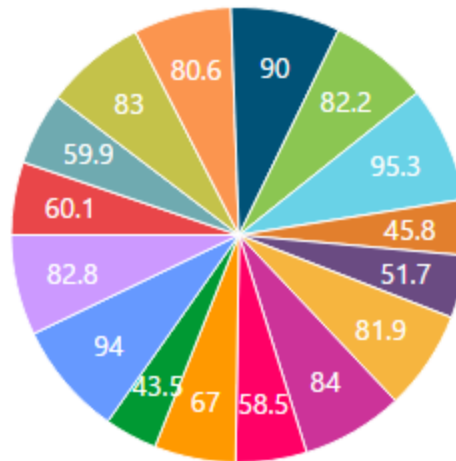
### Position and Align the Legend

By using the [position](#) option, you can position the legend at *left*, *right*, *top* or *bottom* of the chart. The legend is positioned at the **bottom** of the chart, by default.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  legend: {
    // ...
    //Place the legend at top of the chart
    position: 'top',
  }
  // ...
});
```

## Internet users (per 100 people) in 2013



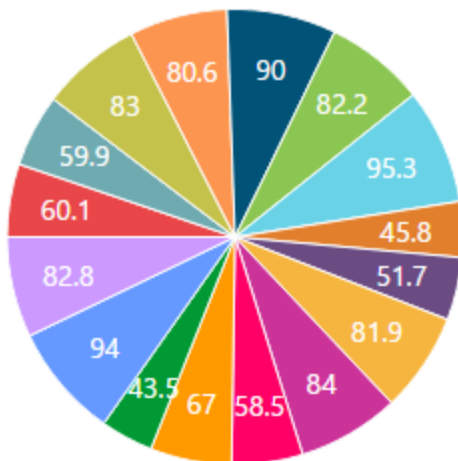
### Legend Alignment

You can align the legend to the *center*, *far* or *near* based on its position by using the [alignment](#) option.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  legend: {
    //...
    //The below two settings will place the legend at the top-right corner of
    //the chart.
    alignment: 'far',
    position: 'top',
  }
  // ...
});
```

## Internet users (per 100 people) in 2013



### Arrange legend items in the rows and columns

You can arrange the legend items horizontally and vertically by using the [rowCount](#) and [columnCount](#) options of the legend.

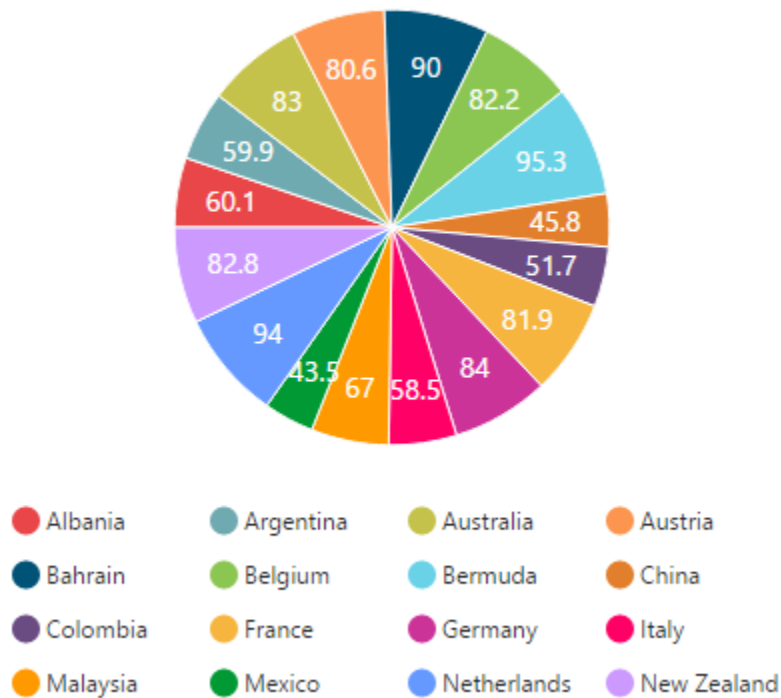
- When only the [rowCount](#) is specified, the legend items are arranged according to the [rowCount](#) and number of columns may vary based on the number of legend items.
- When only the [columnCount](#) is specified, the legend items are arranged according to the [columnCount](#) and number of rows may vary based on the number of legend items.
- When both the options are specified, then the one which has higher value is given preference. For example, when the [rowCount](#) is 4 and [columnCount](#) is 3, legend items are arranged in 4 rows.
- When both the options are specified and have the same value, the preference is given to the [columnCount](#) when it is positioned at the top/bottom position. The preference is given to the [rowCount](#) when it is positioned at the left/right position.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  legend: {
    //Arrange legend items in 4 rows and approximately 4 columns. Column
    //couldn't may vary based on number of items.
    rowCount: 4,
    columnCount: 4
  }
})
```

```
// ...
});
```

Internet users (per 100 people) in 2013



## Customization

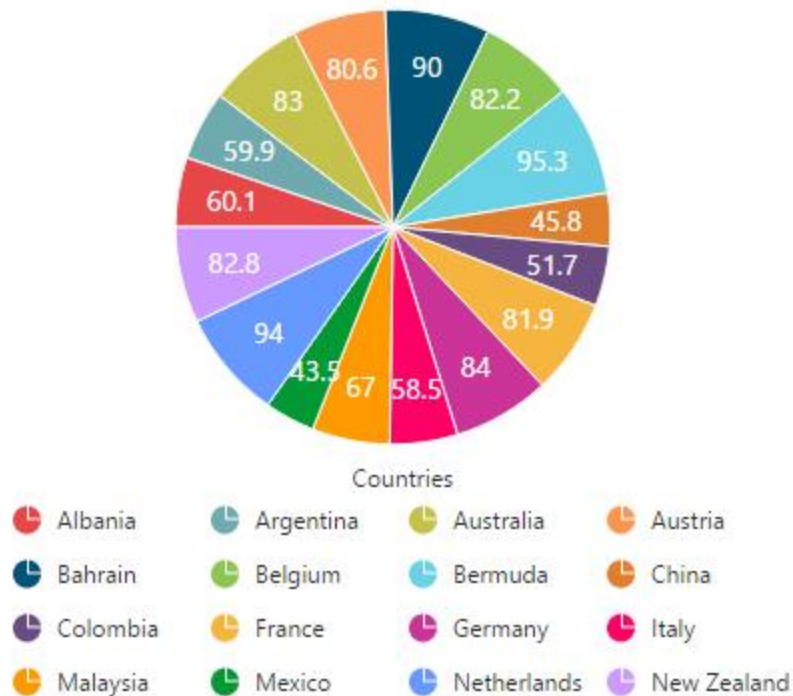
### Legend shape

To change the legend icon shape, you have to specify the shape in the [shape](#) property of the legend. When you want the legend icon to display the prototype of the series, you have to set the **seriesType** as shape.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  legend: {
    //...
    //Change legend shape
    shape: 'seriesType',
  }
  // ...
});
```

Internet users (per 100 people) in 2013



#### Legend items size and border

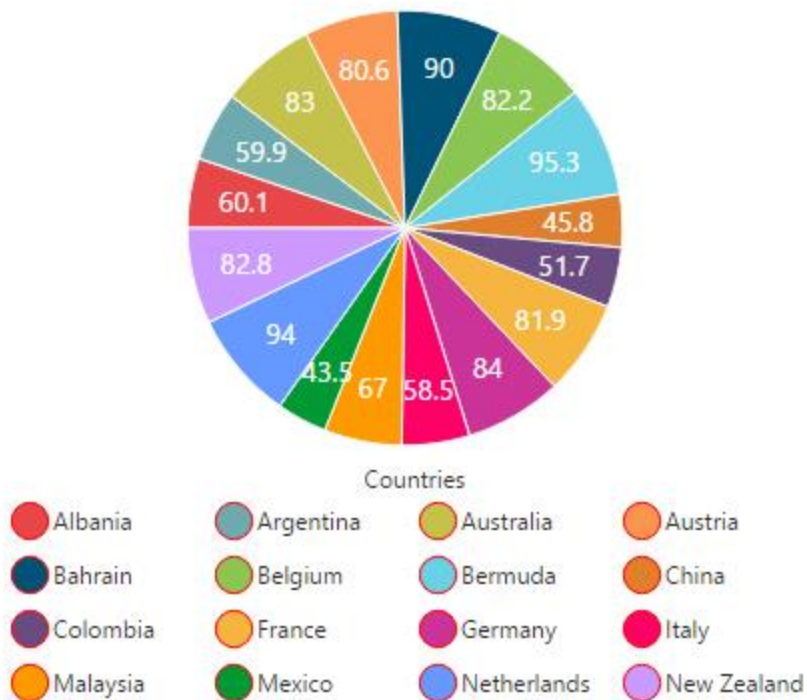
You can change the size of the legend items by using the [itemStyle.width](#) and [itemStyle.height](#) options. To change the legend item border, use [border](#) option of the legend itemStyle.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  legend: {
    //...
    //Change legend items border, height and width
    itemStyle: {width: 13, height: 13, border: { color: "#FF0000", width: 1 } },
  }
  // ...
});
```



Internet users (per 100 people) in 2013



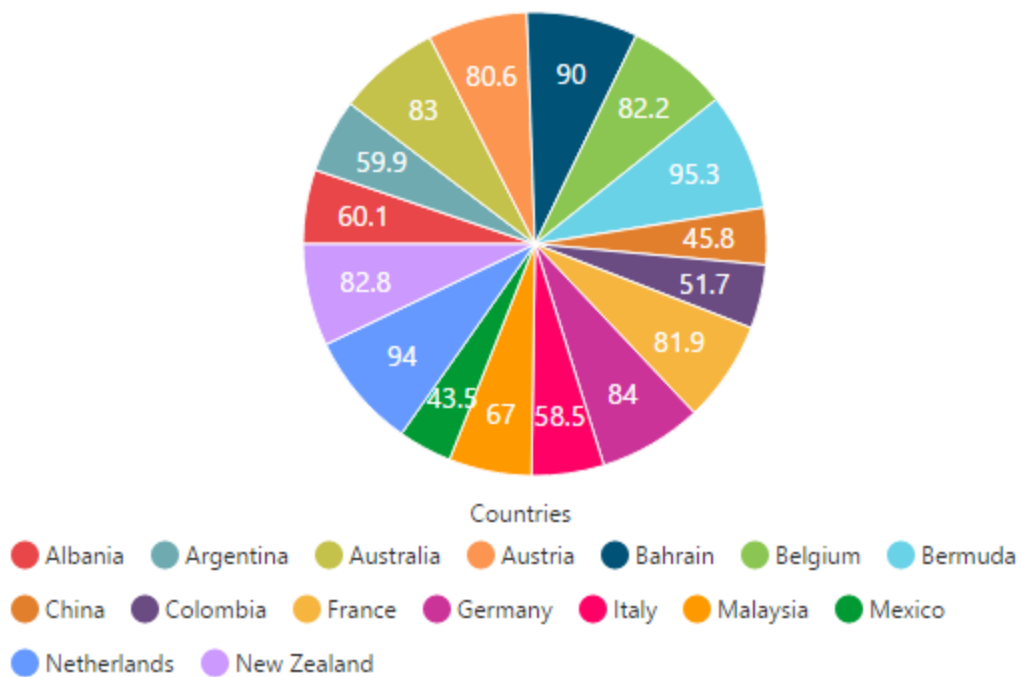
#### Legend size

By default, legend takes 20% of the **height** horizontally when it was placed on the top or bottom position and 20% of the **width** vertically while placing on the left or right position of the chart. You can change this default legend size by using the [size](#) option of the legend.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  legend: {
    //...
    //Change legend size
    size:{width: '550', height: '100'}
  }
  // ...
});
```

Internet users (per 100 people) in 2013



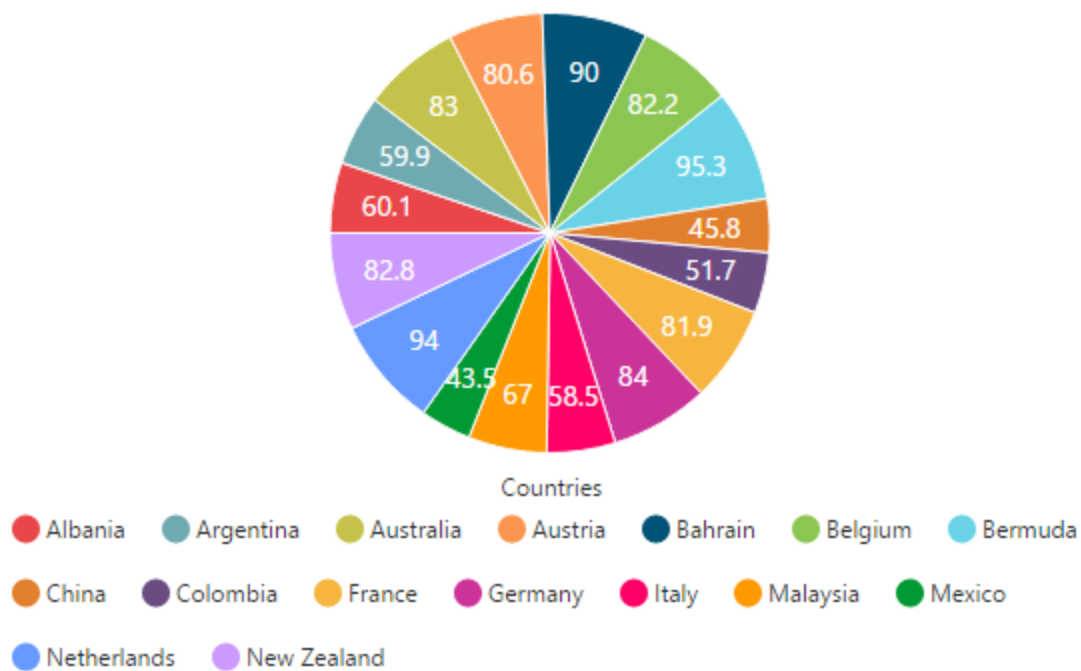
### Legend Item Padding

You can control the spacing between the legend items by using the [itemPadding](#) option of the legend. The default value is 10.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  legend: {
    //...
    //Add space between each legend item
    itemPadding: 15,
  }
  // ...
});
```

Internet users (per 100 people) in 2013



### Legend border

You can customize the legend border by using the [border](#) option in the legend.

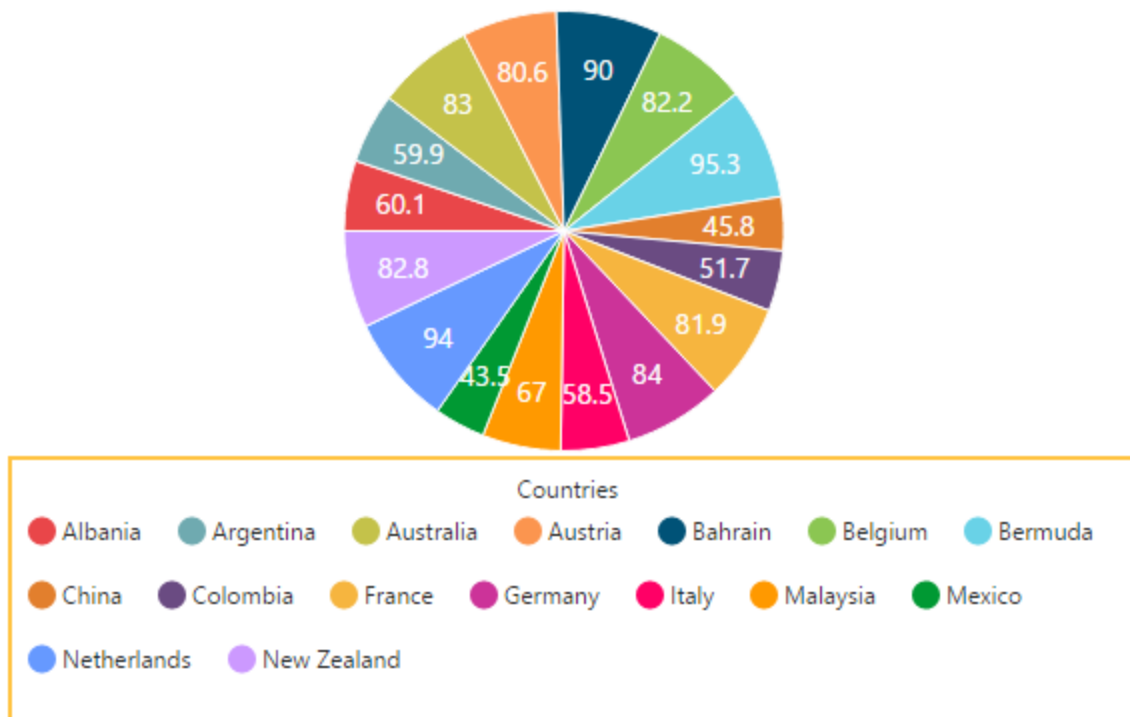
### JAVASCRIPT

```

var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  legend: {
    //...
    //Set border color and width to legend
    border: {color: "#FFC342", width: 2},
  }
  // ...
});

```

Internet users (per 100 people) in 2013



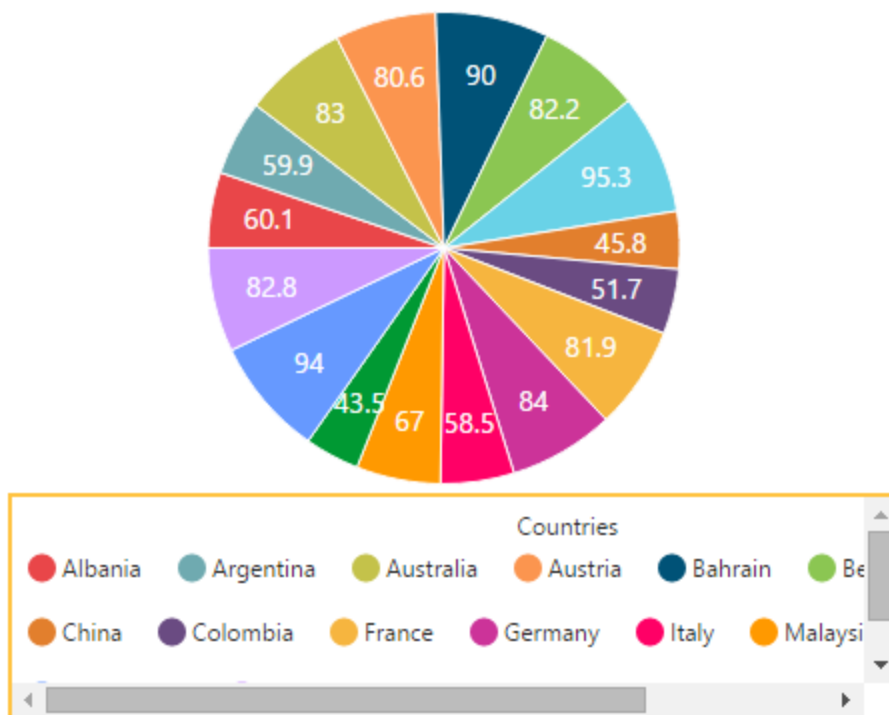
### Scrollbar for legend

You can enable or disable the legend scrollbar by using the [enableScrollbar](#) option of the legend. When you disable the scrollbar option, the legend does not consider the [default size](#) and chart draws in the remaining space. If you have specified the **size** to the legend with the scrollbar disabled, then the legends beyond this limit will get clipped. The default value of [enableScrollbar](#) option is **true**.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  legend: {
    //...
    //Enable scrollbar option in for legend
    enableScrollbar: true,
    size:{width: '430', height: '80'},
  }
  // ...
});
```

Internet users (per 100 people) in 2013



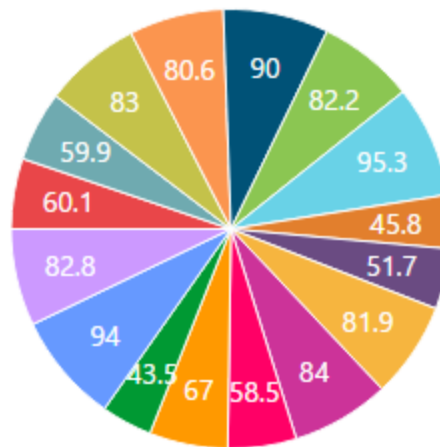
#### Customize the legend text

To customize the legend item text and title you can use the [legend.font](#) and [legend.title](#) options. You can change the legend title alignment by using the [textAlignment](#) option of the legend title.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  legend: {
    //...
    //Customize the legend item text
    font: { fontFamily: 'Segoe UI', fontStyle: 'Normal', fontWeight: 'Bold',
      size: '15px' },
    title: {
      //...
      textAlignment: "center",
      //Customize the legend title text
      font: { fontFamily: 'Segoe UI', fontStyle: 'Italic',
        fontWeight: 'Bold', size: '12px' },
    }
  },
  // ...
});
```

Internet users (per 100 people) in 2013



## Countries

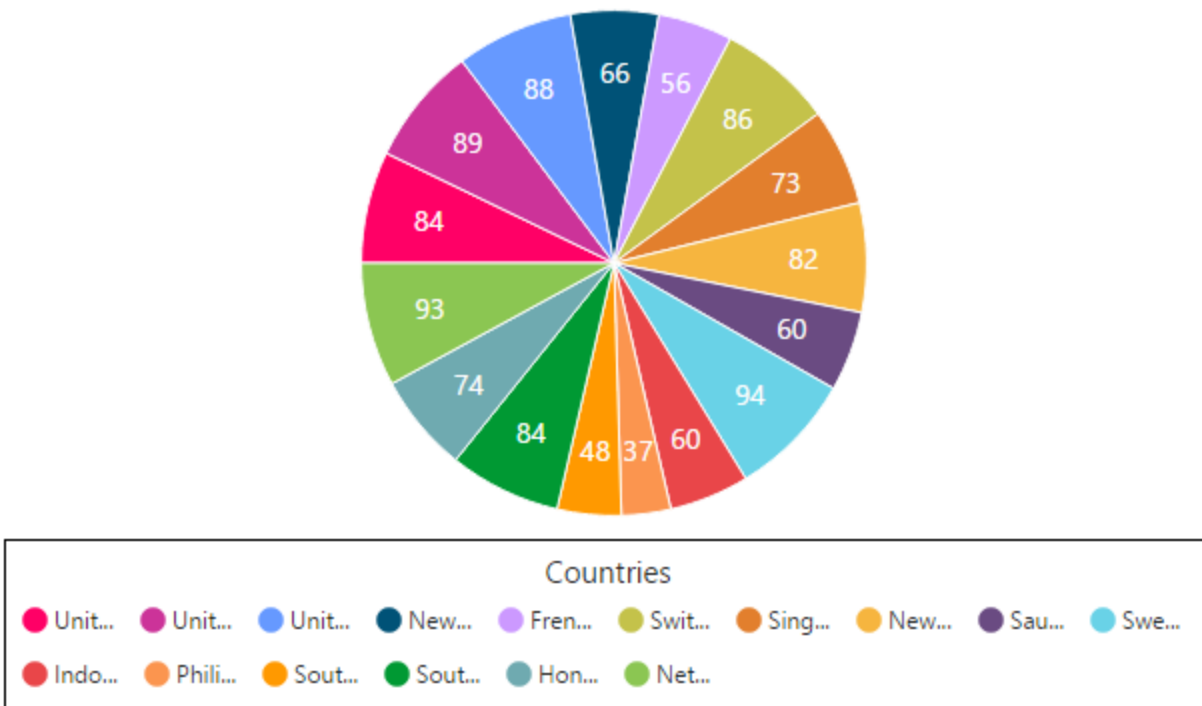
*LegendItems Text Overflow***Trim**

You can trim the legend item text when its width exceeds the [legend.textWidth](#), by specifying [textOverflow](#) as **"trim"**. The original text will be displayed on mouse hover.

**JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  legend: {
    //trim the legend text
    textOverflow: 'trim',
    textWidth: 34
  }
  // ...
});
```

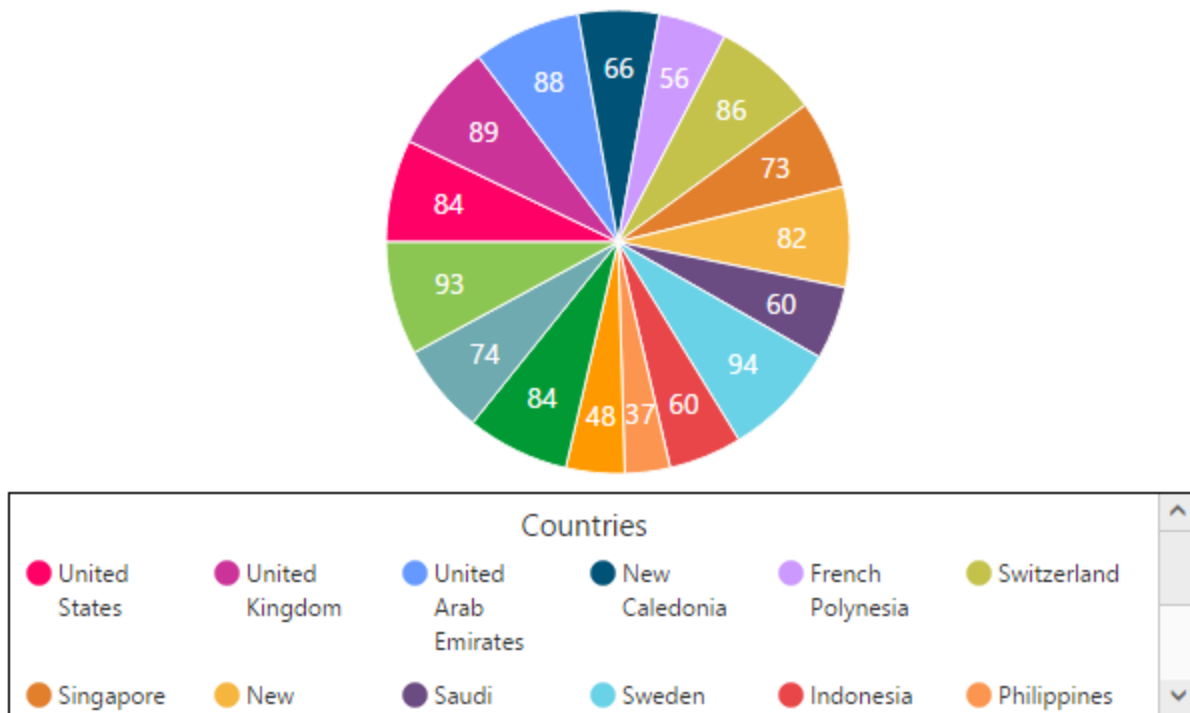
Internet users (per 100 people) in 2013



### Wrap

By specifying [textOverflow](#) as “**wrap**”, you can wrap the legend text by word.

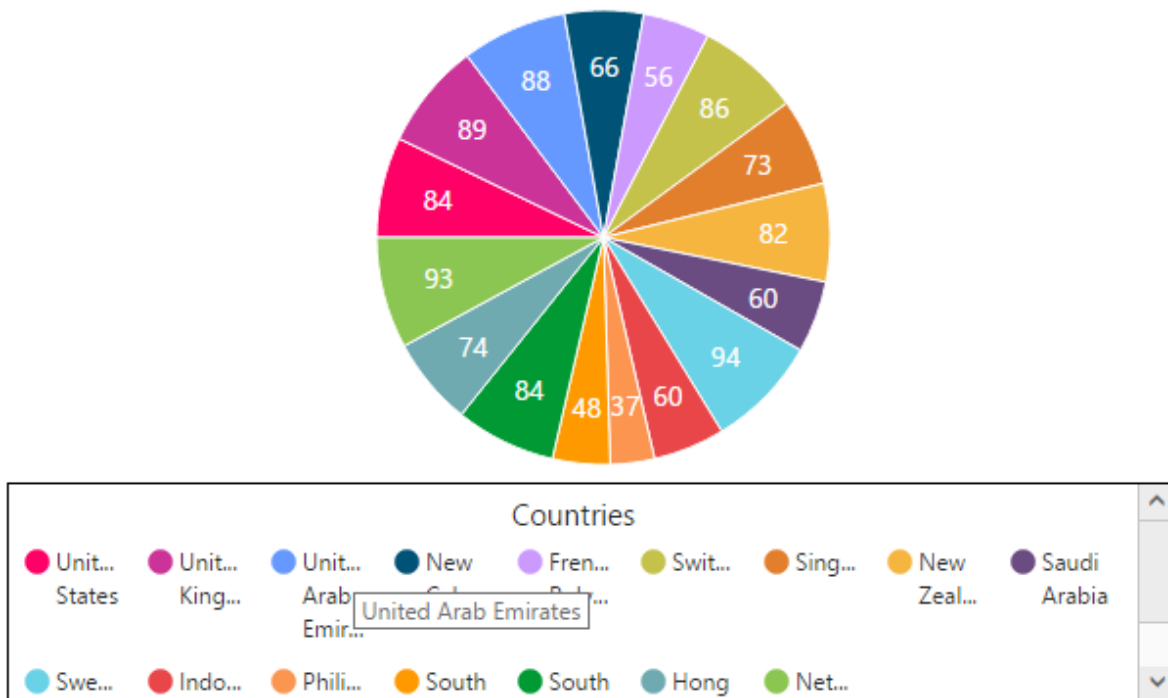
Internet users (per 100 people) in 2013

**WrapAndTrim**

You can wrap and trim the legend text by specifying [textOverflow](#) as **"wrapAndTrim"**. The original text will be displayed on mouse hover.



Internet users (per 100 people) in 2013



### Handle the legend item clicked

You can get the legend item details such as *index*, *bounds*, *shape* and *series* by subscribing the [legendItemClick](#) event on the chart. When the legend item is clicked, it triggers the event and returns the [legend information](#).

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  legend: {
    //...
  },
  //Subscribe the legend item click event
  legendItemClick: "onLegendClicked",
  //...
});
function onLegendClicked(sender) {
  //Get legend item details on legend item click.
  var legendItem = sender.data;
}
```

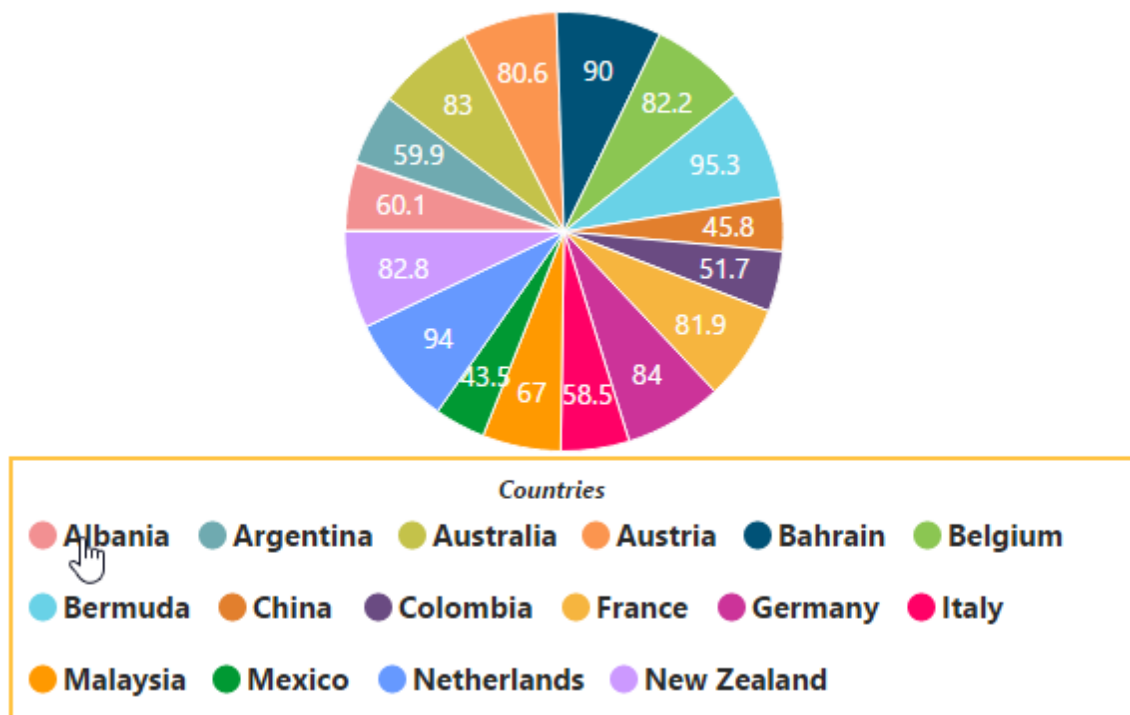
### Series selection on legend item click

You can select a specific series or point while clicking on the corresponding legend item through disabling the [toggleSeriesVisibility](#) option of the legend. The default value of `toggleSeriesVisibility` option is **true**. To customize the series selection refer to the series [selection](#).

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  legend: {
    //...
    //Disable series collapsing on legend item clicked
    toggleSeriesVisibility: false,
  },
  //...
});
```

Internet users (per 100 people) in 2013



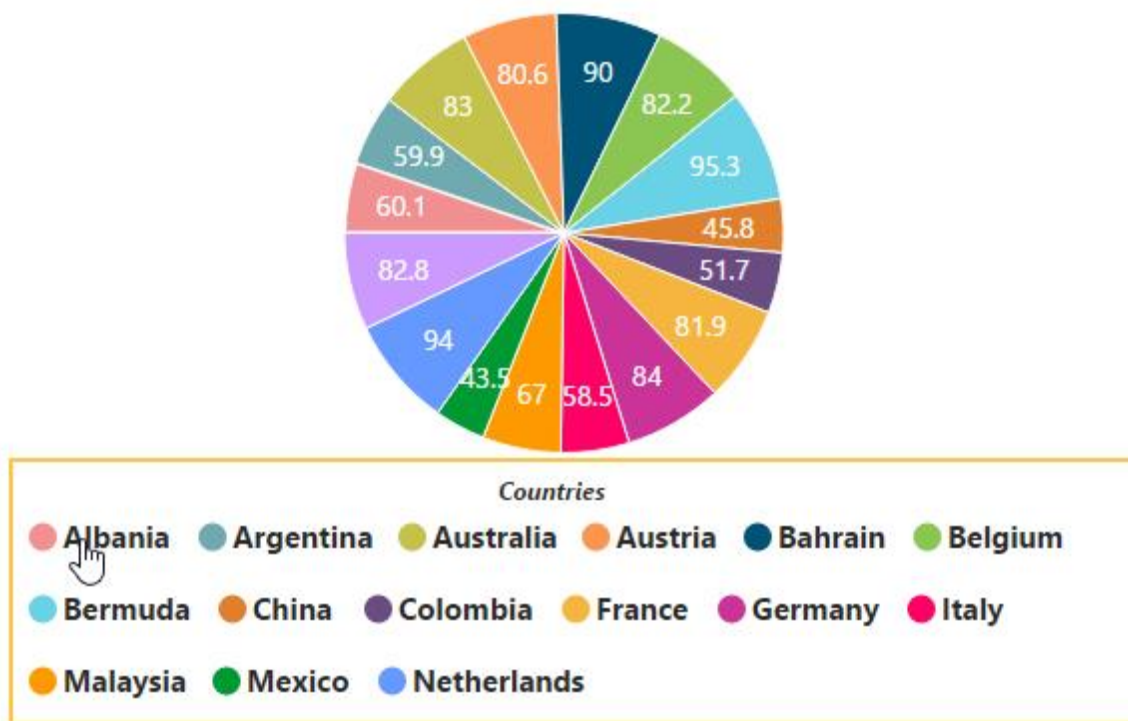
### Collapsing legend item

You can collapse the specific series/point legend item displaying in the chart, by setting the [visibleOnLegend](#) as "hidden" in the point or series.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  //Initializing Series
  series:[{
    points: [{ x: 'Albania', y: 60.1 },
    //...
    //Collapse the point's legend item in the legend collection
    { x: 'New Zealand', y: 82.8, visibleOnLegend:'hidden' } ]
  },
  legend: { visible: true }
});
```

Internet users (per 100 people) in 2013



### Empty Points

The Data points that uses the **null** or **undefined** as value are considered as empty points. Empty data points are ignored and not plotted in the Chart. When the data is provided by using the [points](#) property, you can set the **isEmpty** to true to specify that the particular point is an empty point.

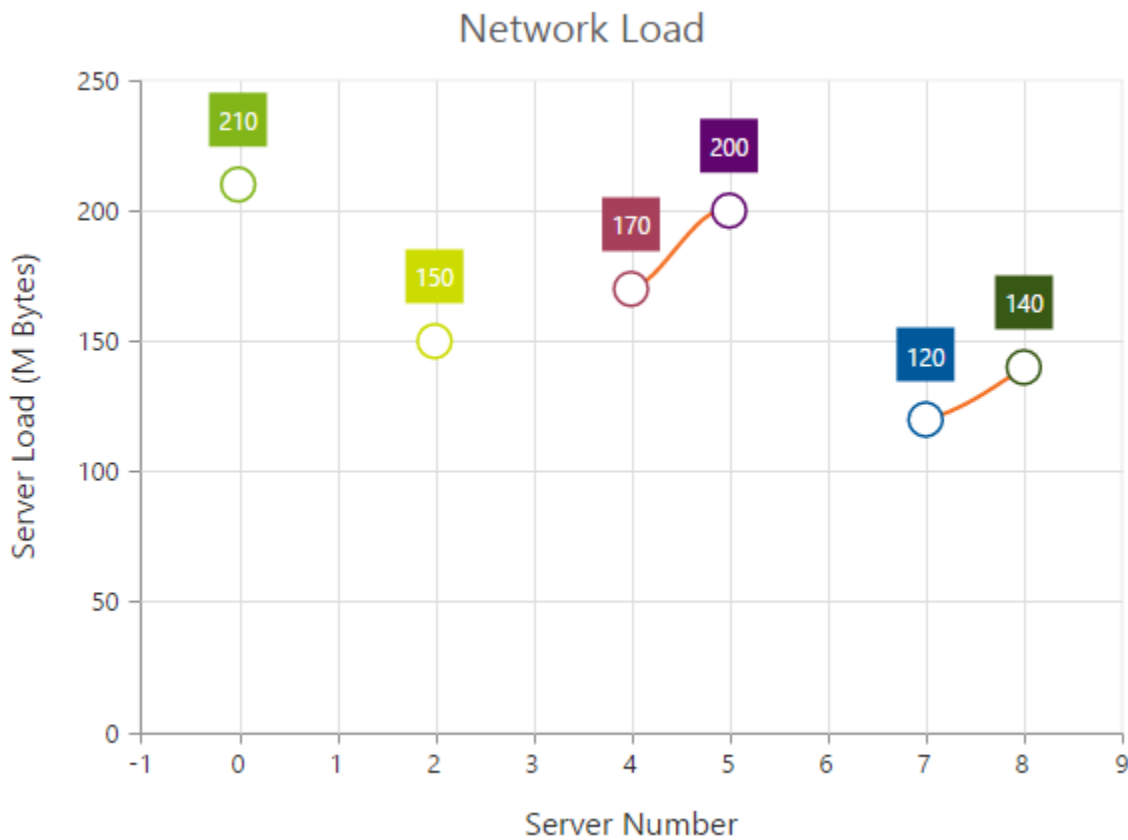
### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
  $(function () {
    var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
      series:[ {
        //Using empty points
        points: [
          { x: 0, y: 210 },
          { x: 1, y: null }, { x: 2, y: 150 },
          { x: 3, y: 180, isEmpty: true },
          { x: 4, y: 170},
          { x: 5, y: 200 },
          { x: 6, y: 140, isEmpty: true },
          { x: 7, y: 120 }, { x: 8, y: 140 }
        ],
      }],
    },
    // ...
  )
}

```

```
});
```

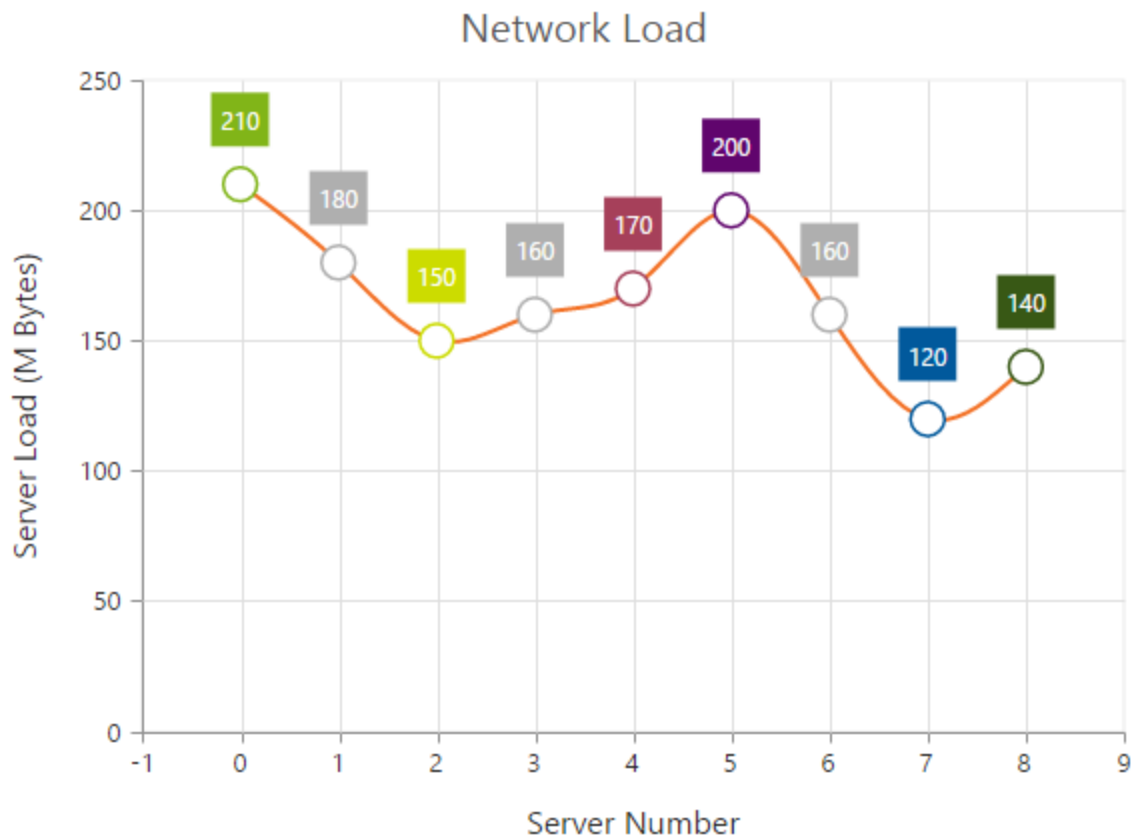


### EmptyPointSettings

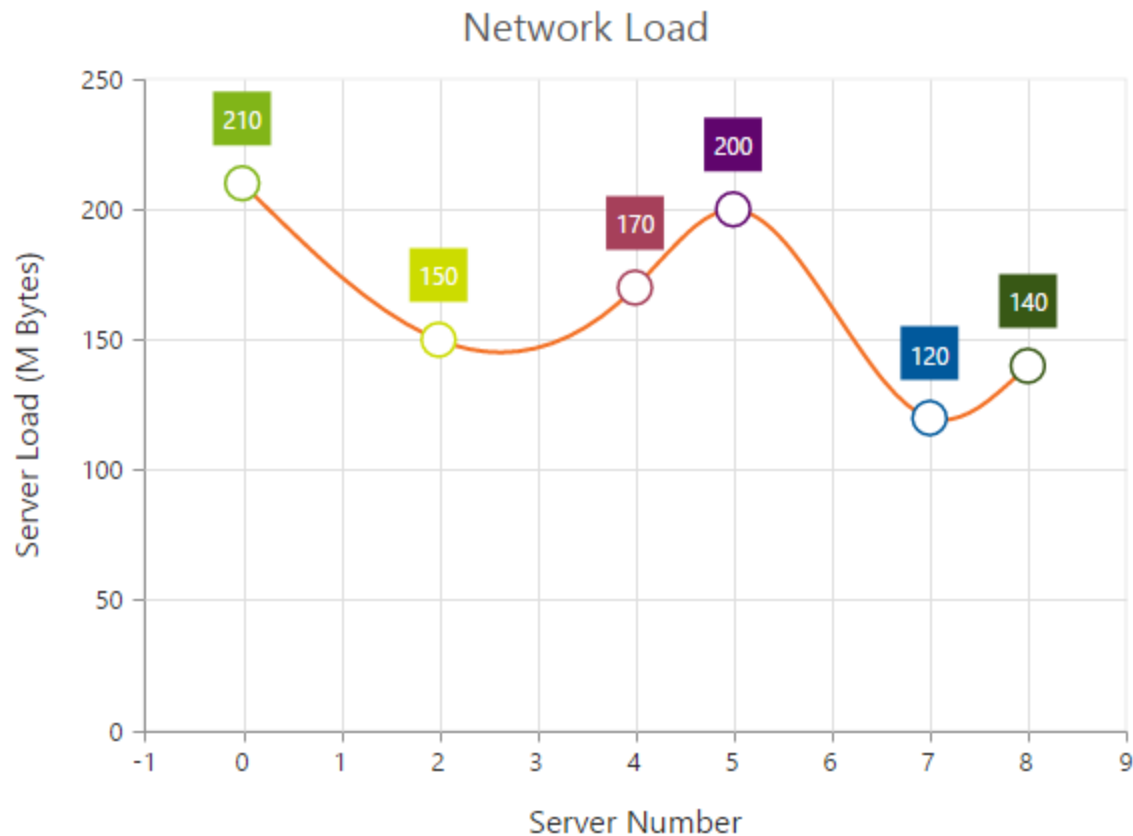
You can customize the empty points visibility and change its [displayMode](#) (*gap, zero and average*) using [emptyPointSettings](#) option.

### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  series:[ {
    //...
    emptyPointSettings: {
      // visible the empty point settings with displayMode as average
      visible: true,
      displayMode : "average"
    },
  },
  ],
  // ...
});
```



If the [visible](#) property of [emptyPointSettings](#) is *false*, then the empty points has been dropped and chart will be rendered without empty points.

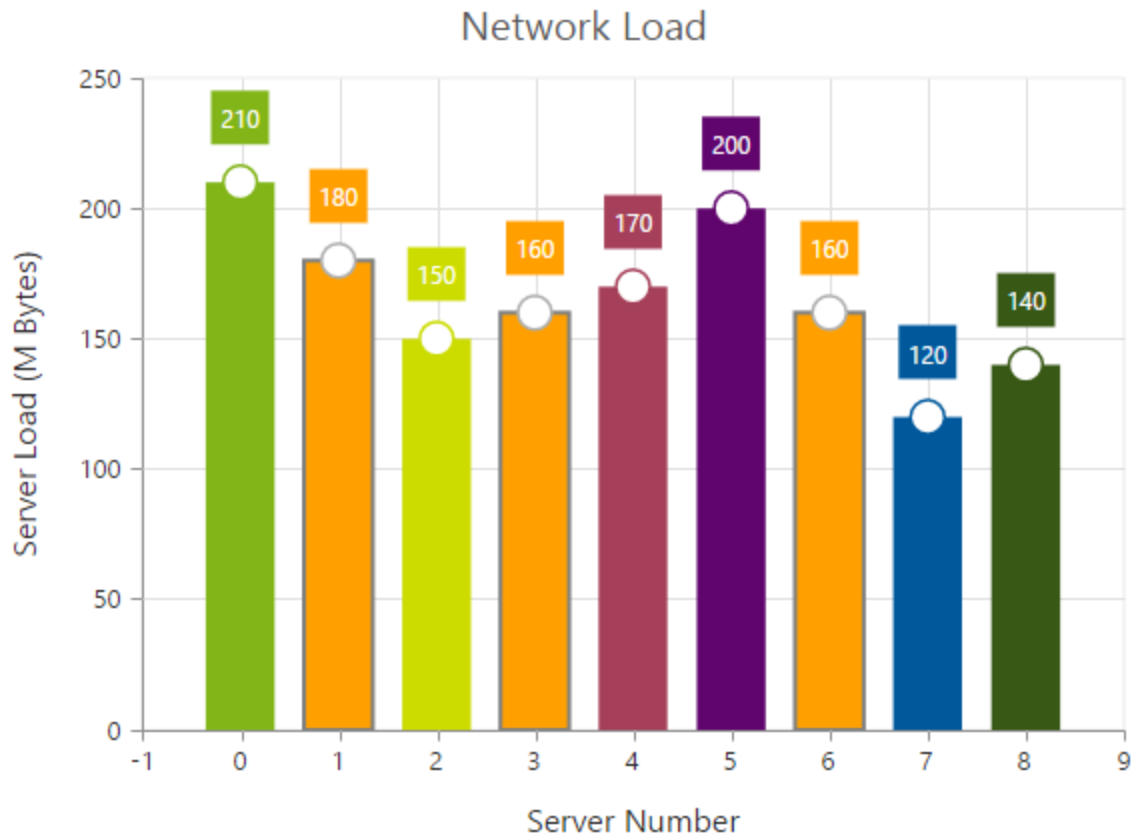


### Customizing Styles

Empty points color and border can be customized using [style](#) property of [emptyPointSettings](#).

### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  series:[{
    //...
    emptyPointSettings: {
      visible: true,
      //Customizing empty points styles
      style: {
        color: "#ffa000",
        border:{
          color: "gray",
          width:2
        }
      }
    },
    // ...
  }];
```



## Chart Title & Subtitle

### Title

By using the title option, you can add the [text](#) as well as customize its [border](#), [background](#) color and [font](#).

### JAVASCRIPT

```

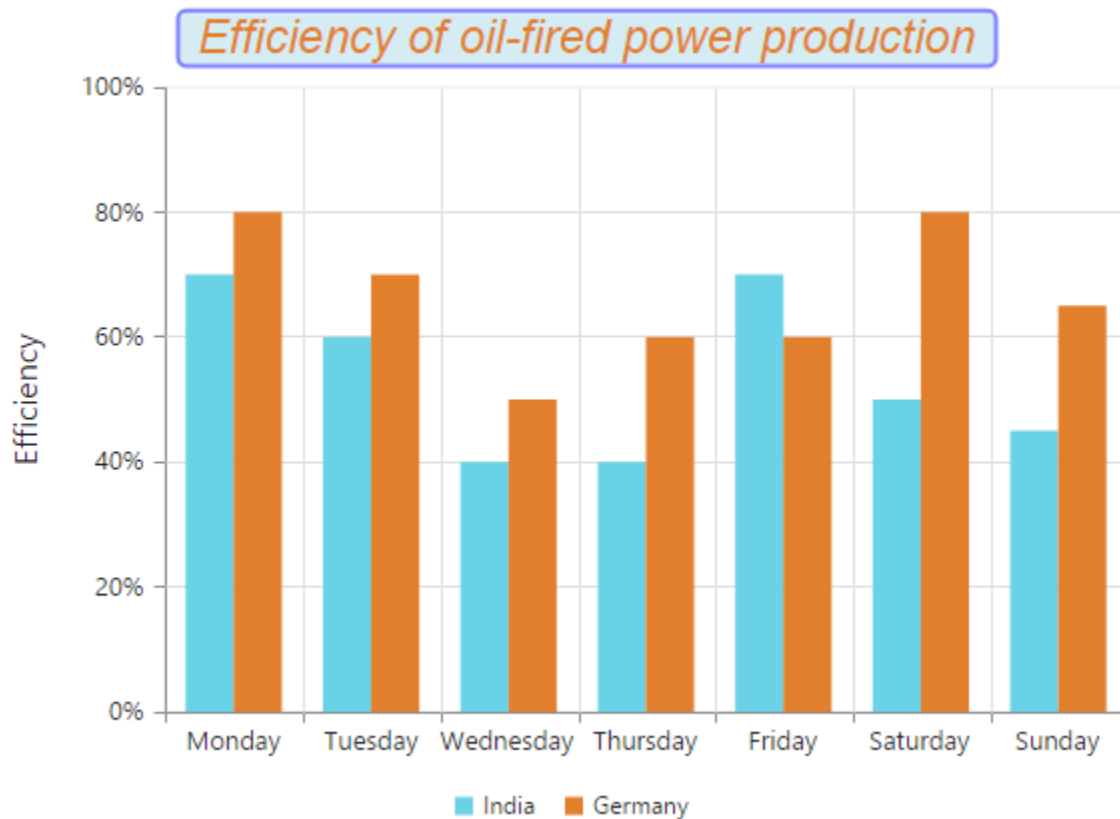
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
$(function () {
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
// ...
title: {
//Adding text to chart title
text: 'Efficiency of oil-fired power production ',
//Change the title text background color
background : "lightblue",
//Customizing Chart title border
border: {
color: "blue",
width: 2,
opacity: 0.5 ,
cornerRadius : 4
},
//Customizing Chart title font
font:{

```

```

opacity: 1,
fontFamily: "Arial",
fontStyle: 'italic',
fontWeight: 'regular',
color: "#E27F2D",
size: '23px'
},
},
// ...
});
});
}

```



We can trim, wrap and wrapAndTrim to the chart title using textOverflow property. The original text will be displayed as tooltip on mouse hover.

### JAVASCRIPT

```

var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
{
// ...
title: {
text: 'Efficiency of oil-fired power production ',
//To enable the title trim, wrap and wrapandtrim
enableTrim: true,
//Setting maximum width to the title

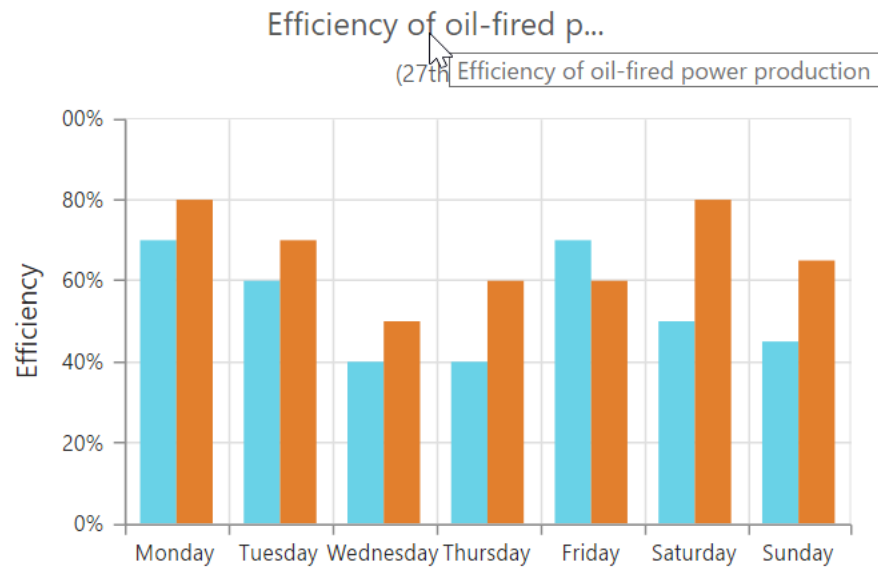
```



```

maximumWidth: 150,
//To trim the title
textOverflow: "trim",
},
// ...
});

```



### Title Alignment

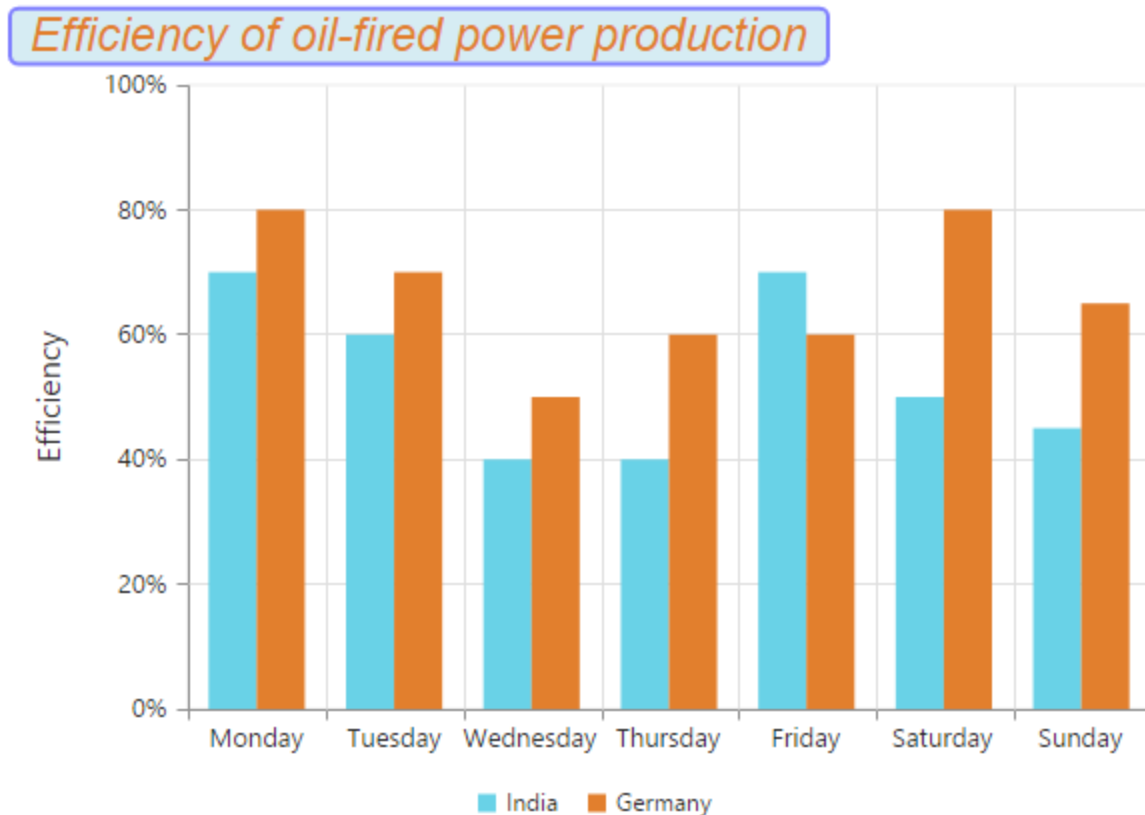
You can change the title alignment to *center*, *far* and *near* by using the [textAlignment](#) property of the chart title.

### JAVASCRIPT

```

var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
// ...
title: {
//Change title text alignment
textAlignment: "near",
//...
}
// ...
});

```



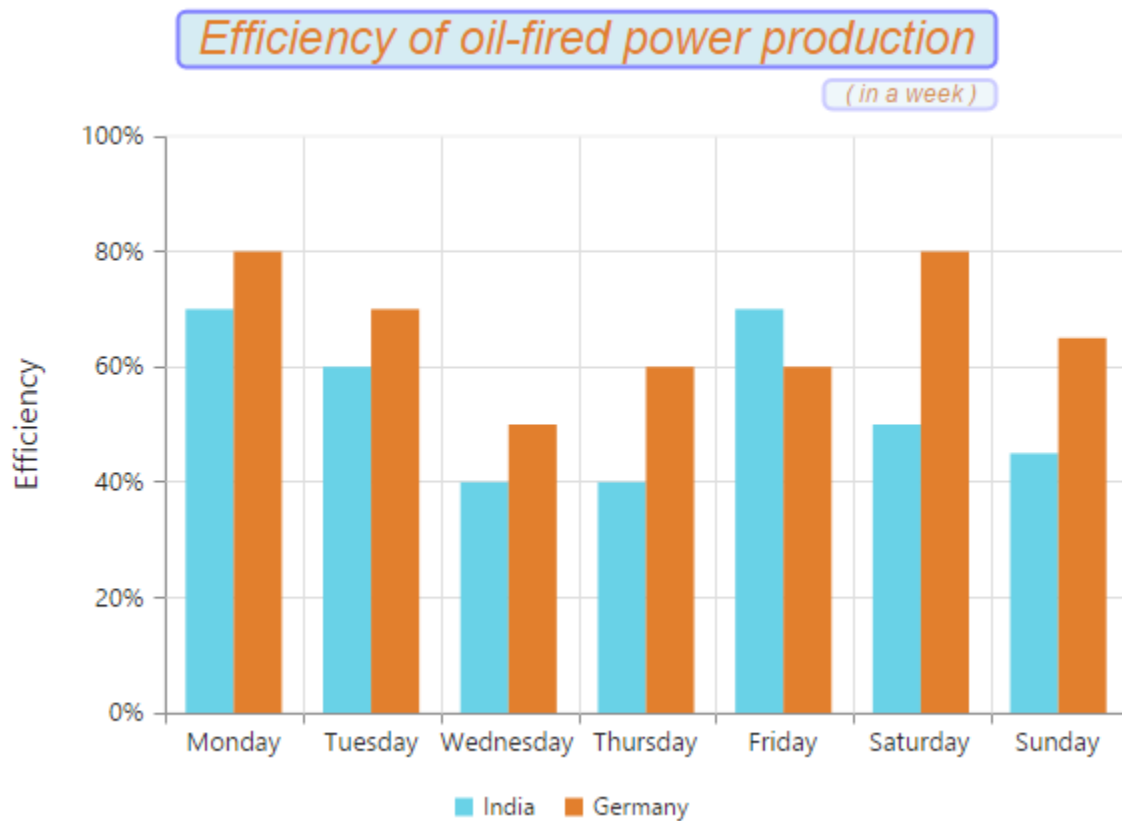
Add Subtitle to the chart

By using the `subTitle` option, you can add the [subTitle](#) to the chart title and customize its [border](#), [background](#) color and [font](#).

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  title: {
    // ...
    subTitle: {
      //Add subtitle to chart title
      text: "( in a week )",
      //Change the title text background color
      background : "lightblue",
      //Customizing Chart subtitle border
      border: {
        color: "blue",
        width: 2,
        opacity: 0.2 ,
        cornerRadius : 4
      },
    },
    //Customizing Chart subtitle font
    font:{
      opacity: 1,
      fontFamily: "Arial",
      fontStyle: 'italic',
```

```
fontWeight: 'regular',
color: "#E27F2D",
size: '12px'
},
}
}
// ...
});
```



We can trim, wrap and wrapAndTrim to the chart subtitle using textOverflow property. The original text will be displayed as tooltip on mouse hover.

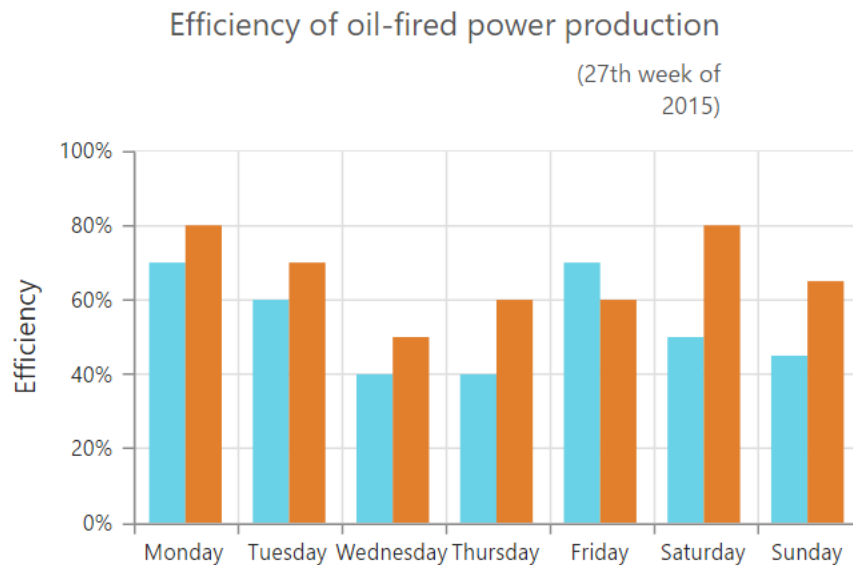
### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
{
// ...
title: {
// ...
subTitle:{
text: '( in a week )',
//To enable the sub-title trim, wrap and wrap and trim
enableTrim: true,
//Setting maximum width to the sub-title
maximumWidth: 50,
//To trim the sub-title
textOverflow: "wrap",
```

```

},
},
// ...
});

```



#### Subtitle Alignment

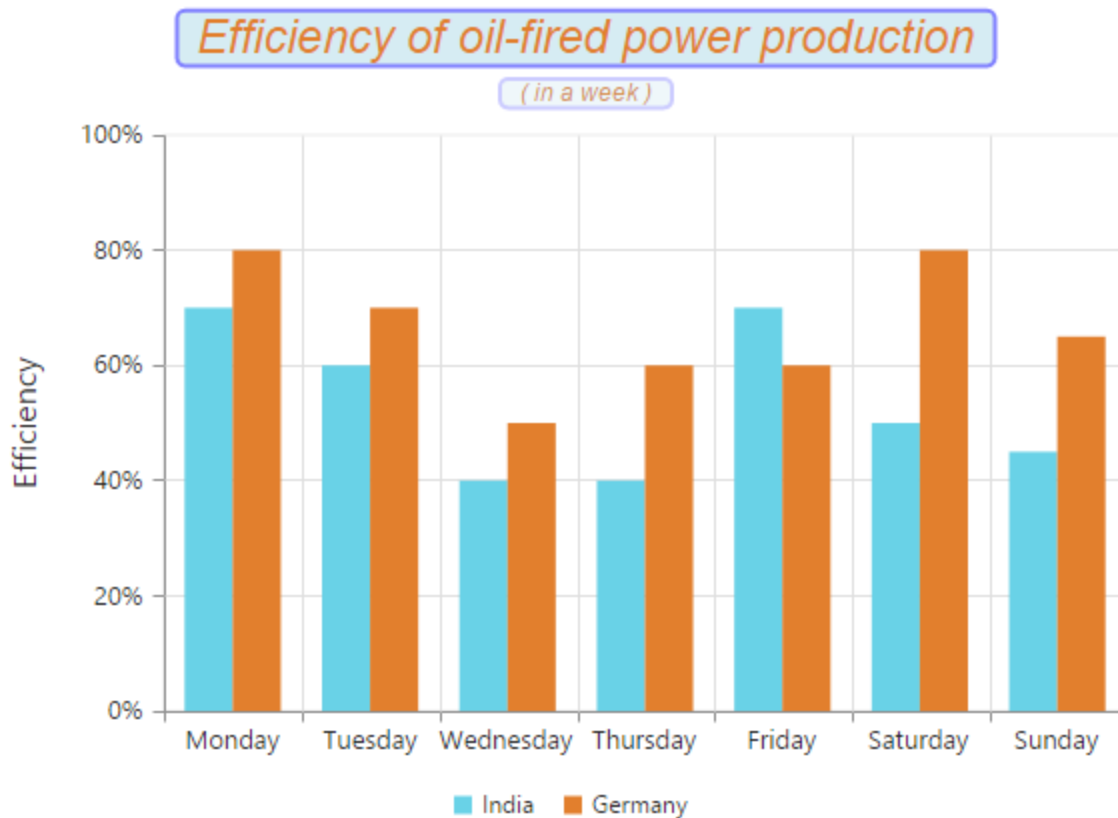
You can change the subtitle alignment to *center*, *far* and *near* by using the [textAlignment](#) property of the subTitle.

#### JAVASCRIPT

```

var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
// ...
title: {
// ...
subTitle:{
//Change subtitle to text alignment
textAlignment: "center",
// ...
}
}
// ...
});

```



## Striplines

EjChart supports horizontal and vertical striplines.

### Horizontal Stripline

You can create horizontal stripline by adding the [stripline](#) in the **vertical axis** and set [visible](#) option to **true**. Striplines are rendered in the specified **start** to **end** range and you can add more than one stripline for an axis.

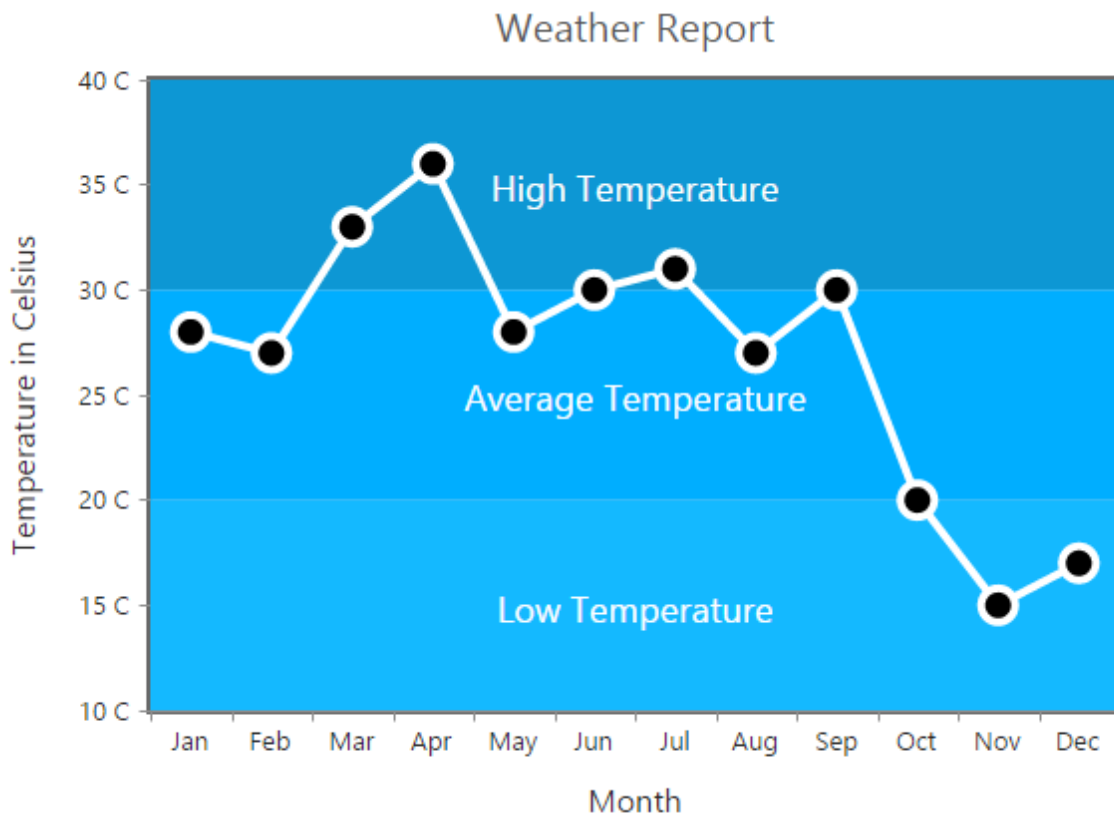
### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
$(function () {
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
// ...
//Initializing Primary Y Axis
primaryYAxis: {
// ...
stripLine: [
//Create horizontal Stripline using vertical Axis
{
//Enable Stripline
visible: true,
start: 30,
end: 40,
},
},
},

```

```
// ...
],
},
// ...
});
});
}
```



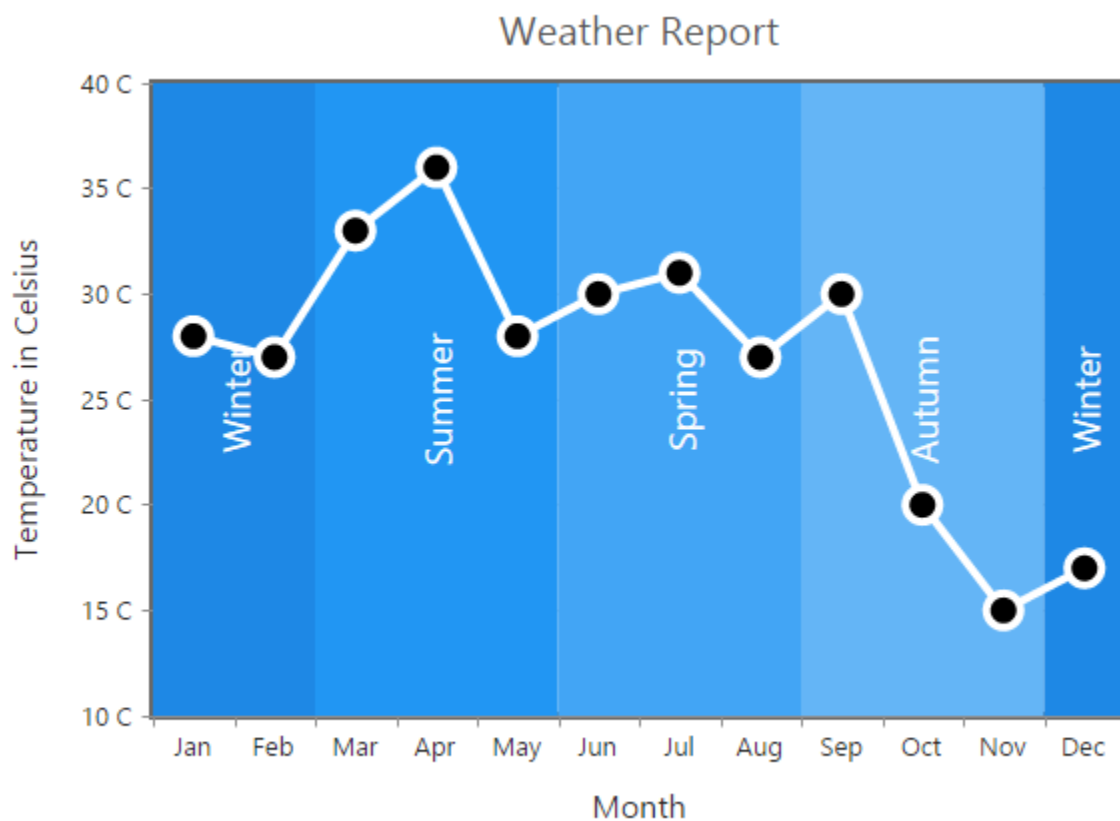
### Vertical Stripline

You can create vertical stripline by adding the [stripline](#) in the **horizontal axis** and set [visible](#) option to **true**.

### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  //Initializing Primary X Axis
  primaryXAxis: {
    // ...
    stripLine: [
      //Create vertical Stripline using horizontal Axis
      {
        //Enable Stripline
        visible: true,
        start: 3,
```

```
end: 7,
},
// ...
],
},
// ...
});
```



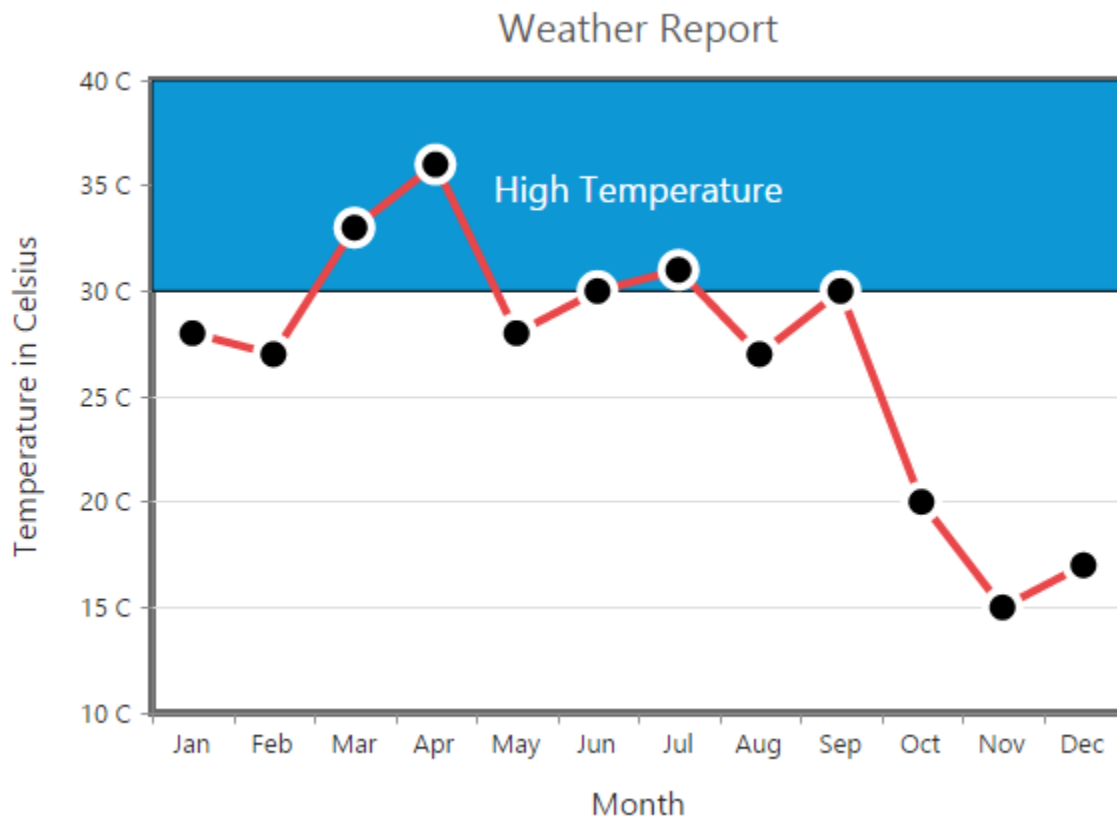
Customize the Text

To customize the stripLine text, use the [text](#) and [font](#) options.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
// ...
//Initializing Primary Y Axis
primaryYAxis: {
// ...
stripLine: [{
//Customize the stripLine text and font styles
text: 'High Temperature',
font: { size: '18px', color: 'white' }
// ...
}],
},
// ...
```

```
});
```



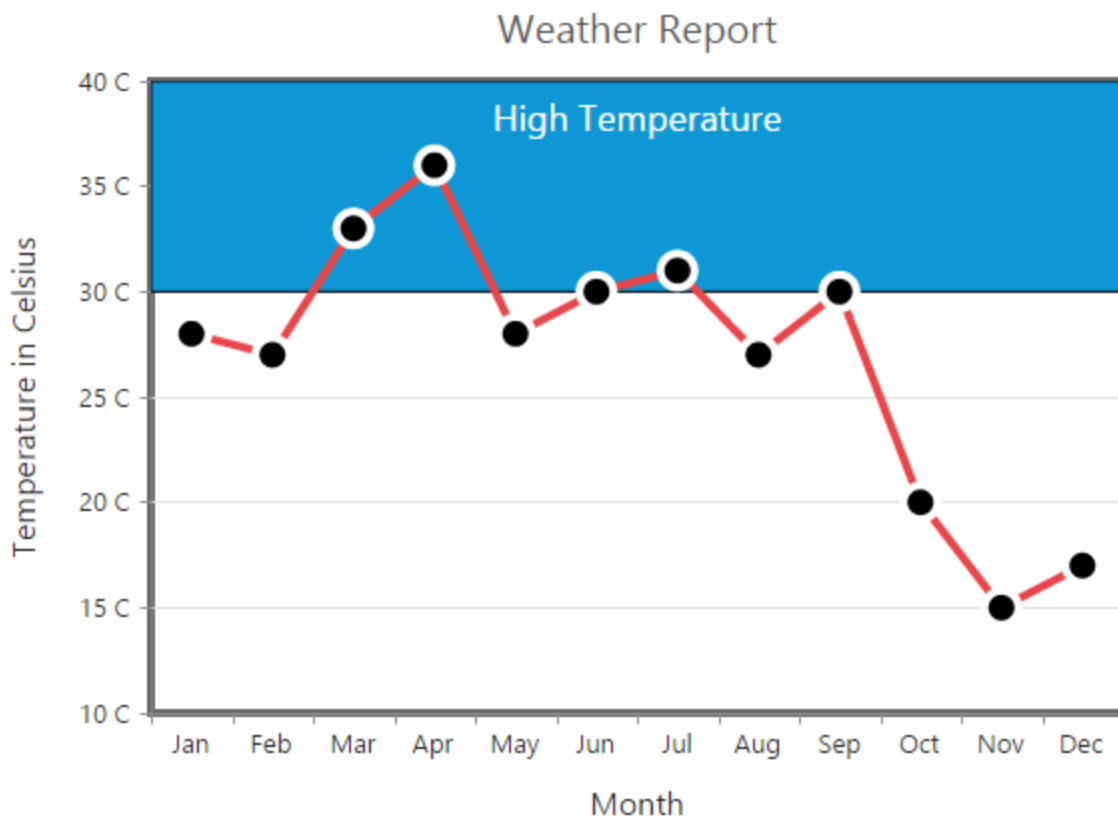
### Text Alignment

Stripline text can be aligned by using the [textAlignment](#) property.

### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  //Initializing Primary Y Axis
  primaryYAxis: {
    // ...
    stripLine: [{
      //Set stripLine text alignment to top position
      textAlignment: 'middletop',
      // ...
    }],
  },
  // ...
});
```



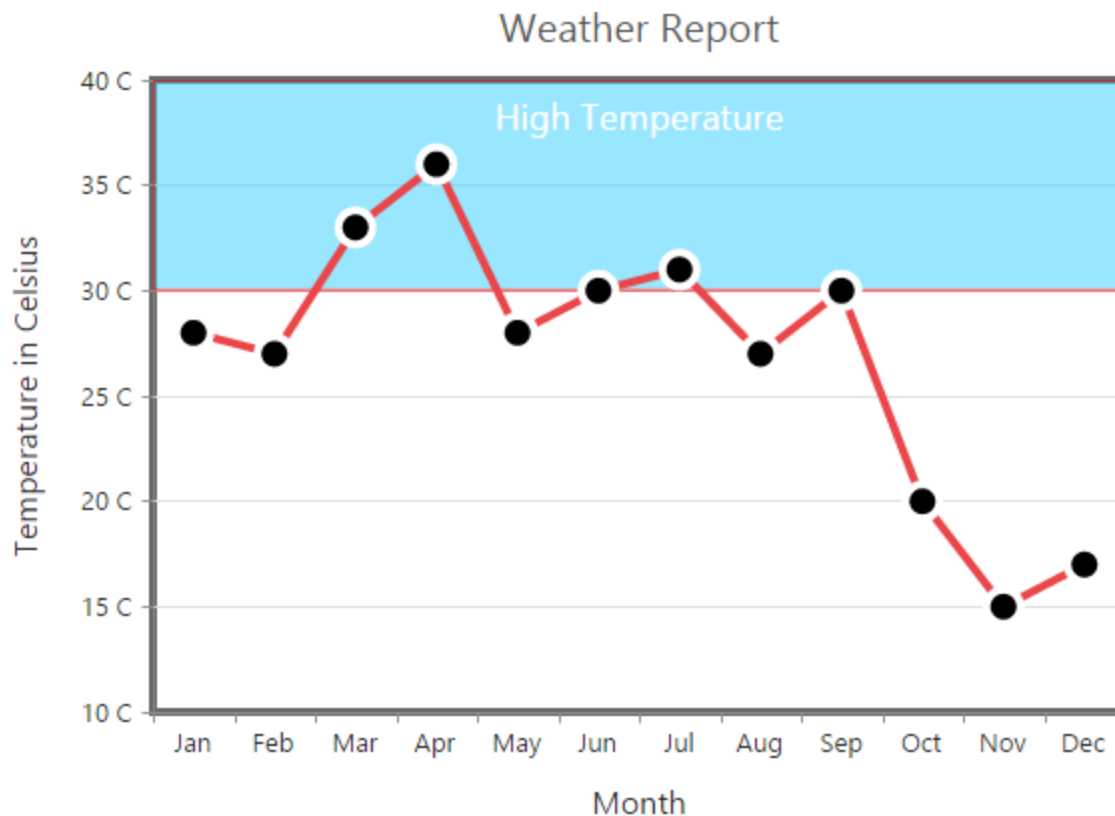


### Customize the Stripline

To customize the stripline styles, use the [color](#), [opacity](#), [borderWidth](#) and [borderColor](#) properties.

### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  //Initializing Primary Y Axis
  primaryYAxis: {
    // ...
    stripLine: [{
      //Customize the Stripline rectangle
      color: '#33CCFF',
      borderWidth: 2,
      opacity: 0.5,
      borderColor: 'red',
      // ...
    }],
  },
  // ...
});
```

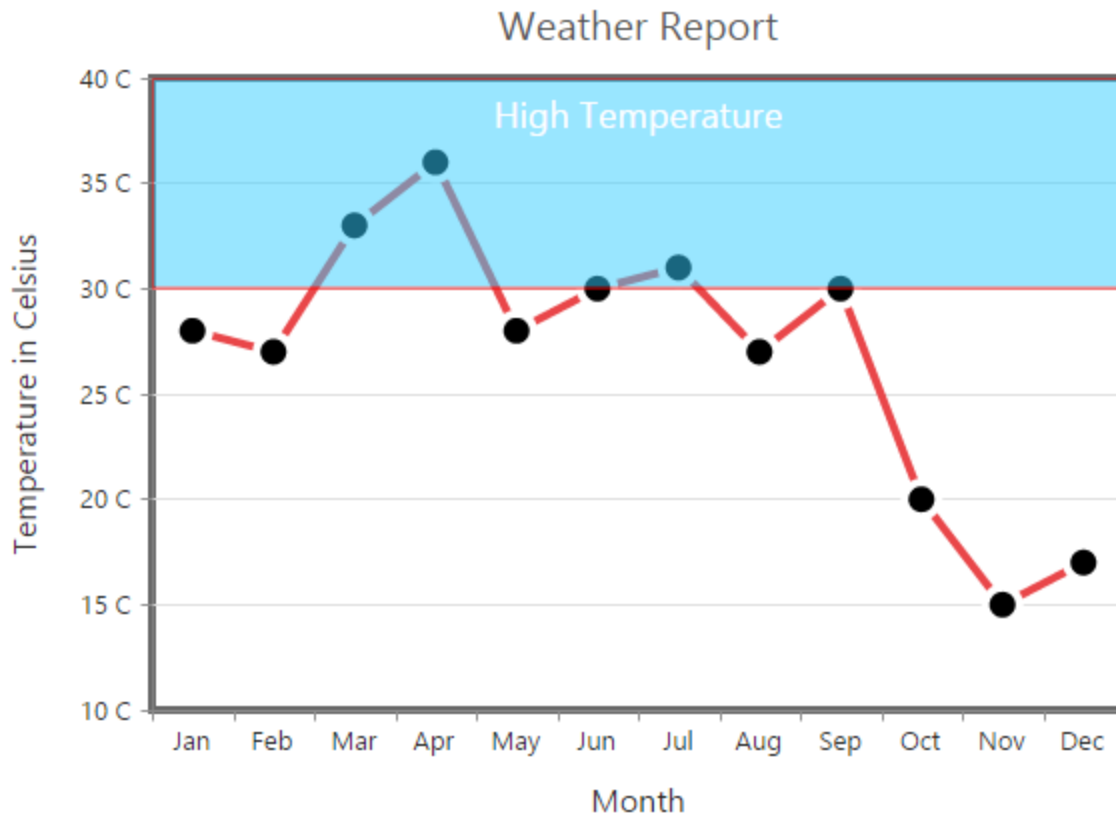


Change the Z-order of the stripline

Stripline [zIndex](#) property is used to display the stripLine either behind or over the series.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  //Initializing Primary Y Axis
  primaryYAxis: {
    // ...
    stripLine: [{
      //Change stripLine zIndex
      zIndex: 'over',
      // ...
    }],
  },
  // ...
});
```



## User Interactions

### Tooltip

*Enable tooltip for data point*

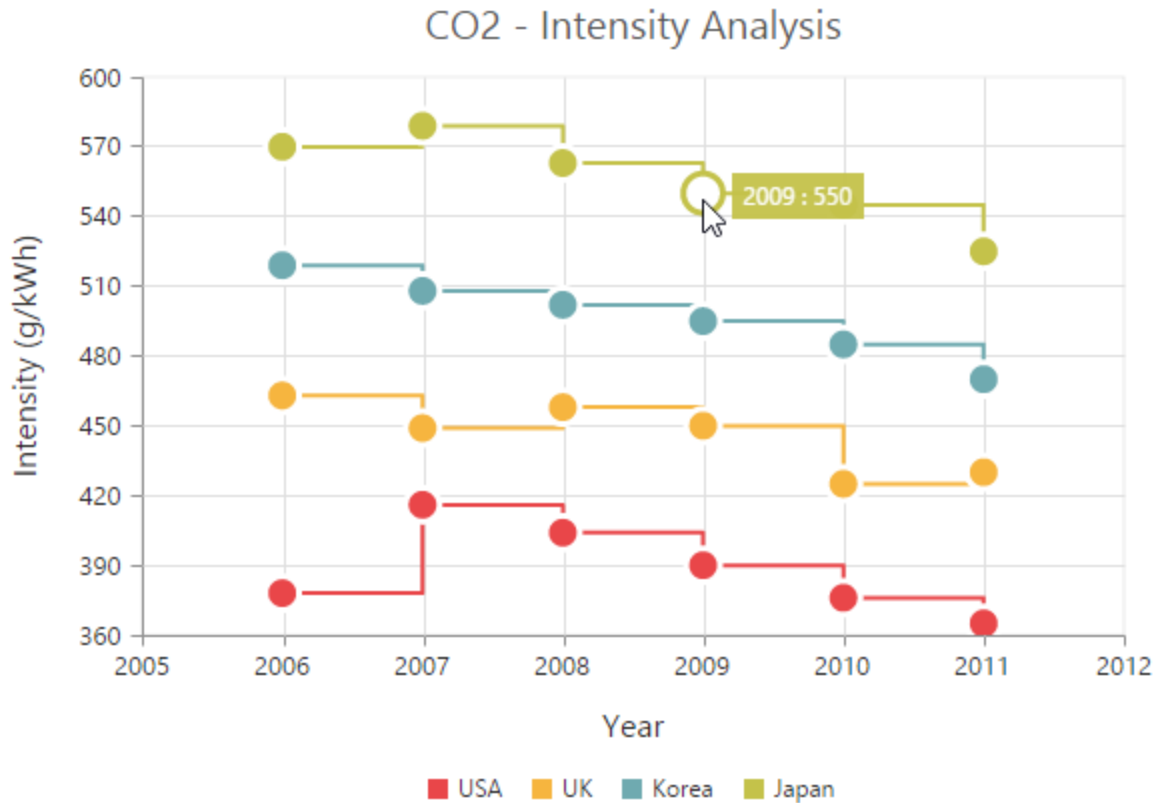
Tooltip for the data points can be enabled by using the [visible](#) option of the [tooltip](#) in the series.

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
$(function () {
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
series: [{
//...
//Enable tooltip for the series
tooltip: {visible: true}
}],
// ...
});
});
}

```



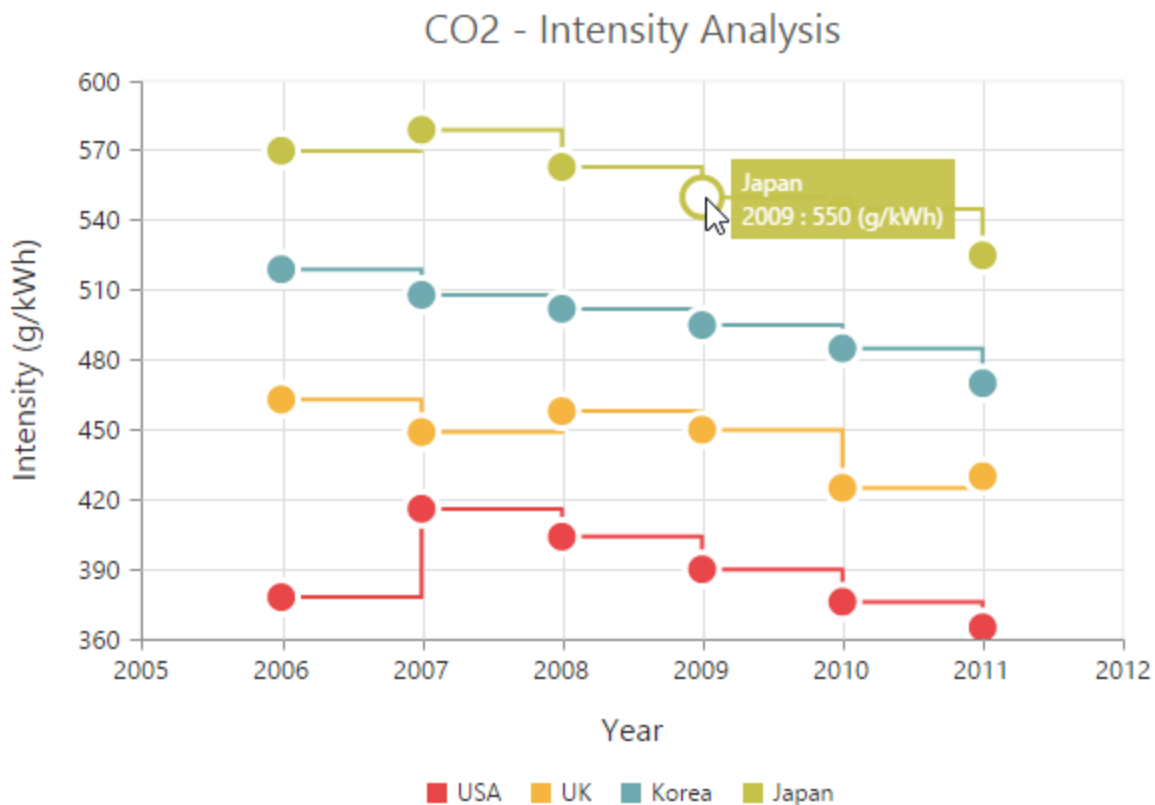
#### Format the tooltip

Tooltip displays data specified by the [format](#) option of the tooltip. The default value of the format option is `#point.x# : #point.y#`. Here, `#point.x#` is the placeholder for x value of the point and `#point.y#` is the placeholder for y value of the point.

You can also use `#series.<optionname>#` as placeholder to display the value of an option in corresponding series and use `#point.<optionname>#` as placeholder to display the value of an option in the corresponding point.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    tooltip: {
      //Displaying tooltip in a format
      format: "#series.name# <br/> #point.x# : #point.y# (g/kWh)"
      // ...
    }
  }],
  // ...
});
```



#### Tooltip Template

HTML elements can be displayed in the tooltip by using the [template](#) option of the tooltip. The template option takes the value of the id attribute of the HTML element. You can use the `#point.x#` and `#point.y#` as place holders in the HTML element to display the x and y values of the corresponding data point.

You can also use `#series.<optionname>#` as place holder to display the value of an option in corresponding series of the tooltip and use `#point.<optionname>#` as place holder to display the value of an option in the corresponding point for which the tooltip is displayed.

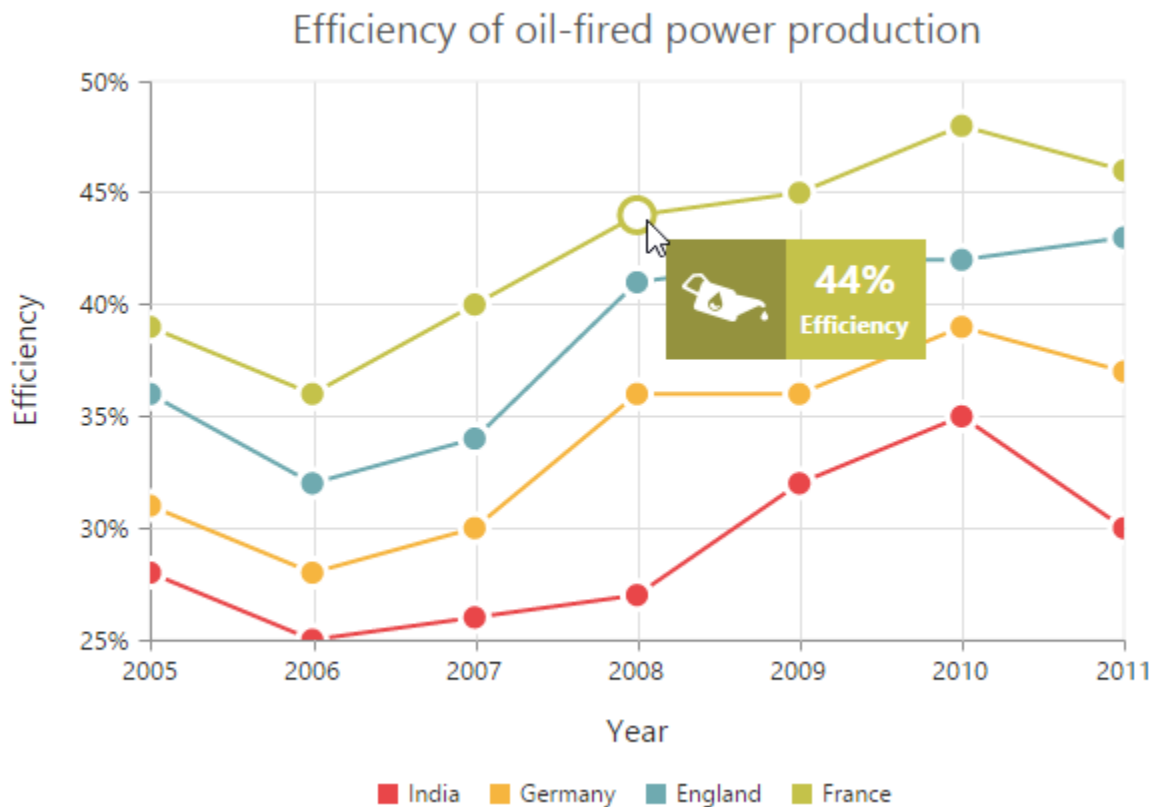
#### HTML

```
<body>
<div id="chartContainer"></div>
<!-- Create Tooltip template here -->
<div id="Tooltip" style="display: none;">
<div id="icon"> <div id="eficon"> </div> </div>
<div id="value">
<div>
<label id="efpercentage">&#160;#point.y#% </label>
<label id="ef">Efficiency </label>
</div>
</div>
</div>
<script>
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
```

```

series: [{
  tooltip: {
    //Set template id to tooltip template
    template: 'Tooltip',
    // ...
  },
},
// ...
});
</script>
</body>

```



#### Tooltip template animation

You can enable animation by setting the [enableAnimation](#) to true. Tooltip animates when the mouse moves from one data point to another point. The [duration](#) property in tooltip specifies the time taken to animate the tooltip. the duration is set to “500ms”, by default.

**Note:** Tooltip is animated only if the template is specified for tooltip.

#### JAVASCRIPT

```

var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    tooltip: {
      //Enable tooltip template animation and set duration time
      enableAnimation: true, duration: "1000ms",

```

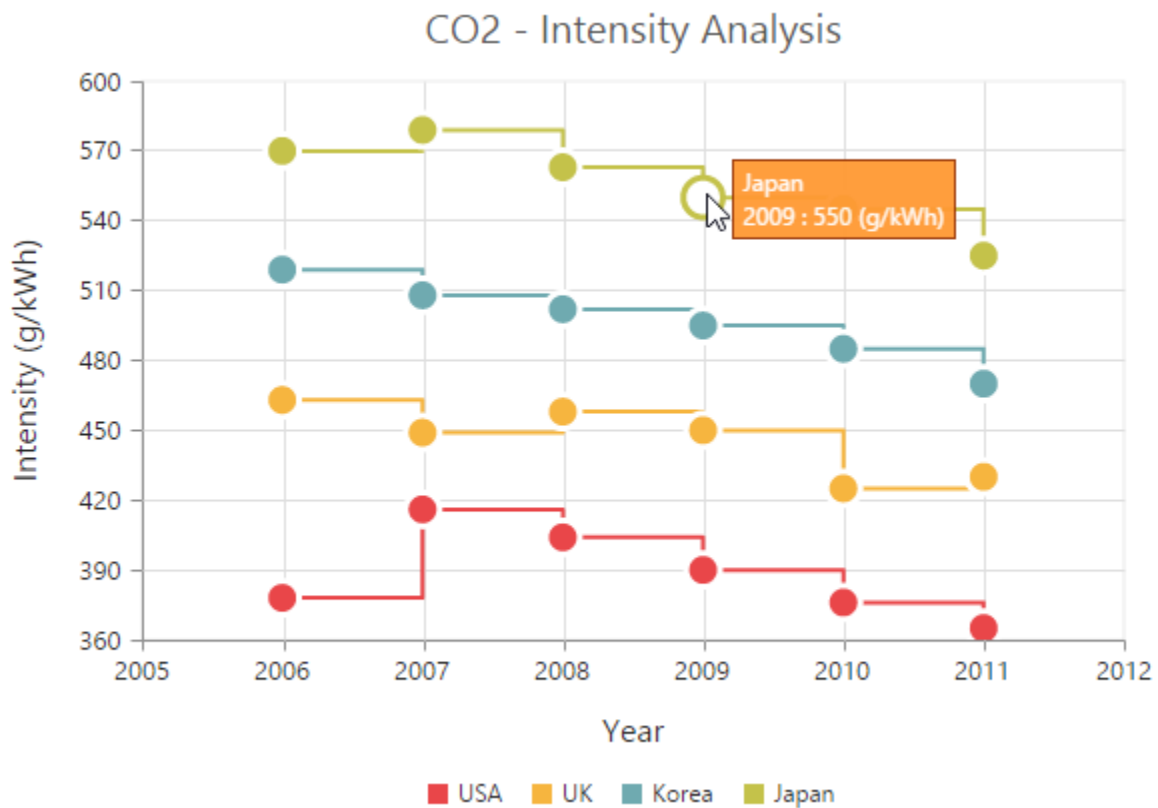
```
// ...
}
}],
// ...
});
```

### Customize the appearance of tooltip

The [fill](#) and [border](#) options are used to customize the background color and border of the tooltip respectively. The [font](#) option in the tooltip is used to customize the font of the tooltip text.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
series: [{
tooltip: {
//Change tooltip color and border
fill: '#FF9933',
border: { width: 1, color: "#993300" }
// ...
}
}],
// ...
});
```

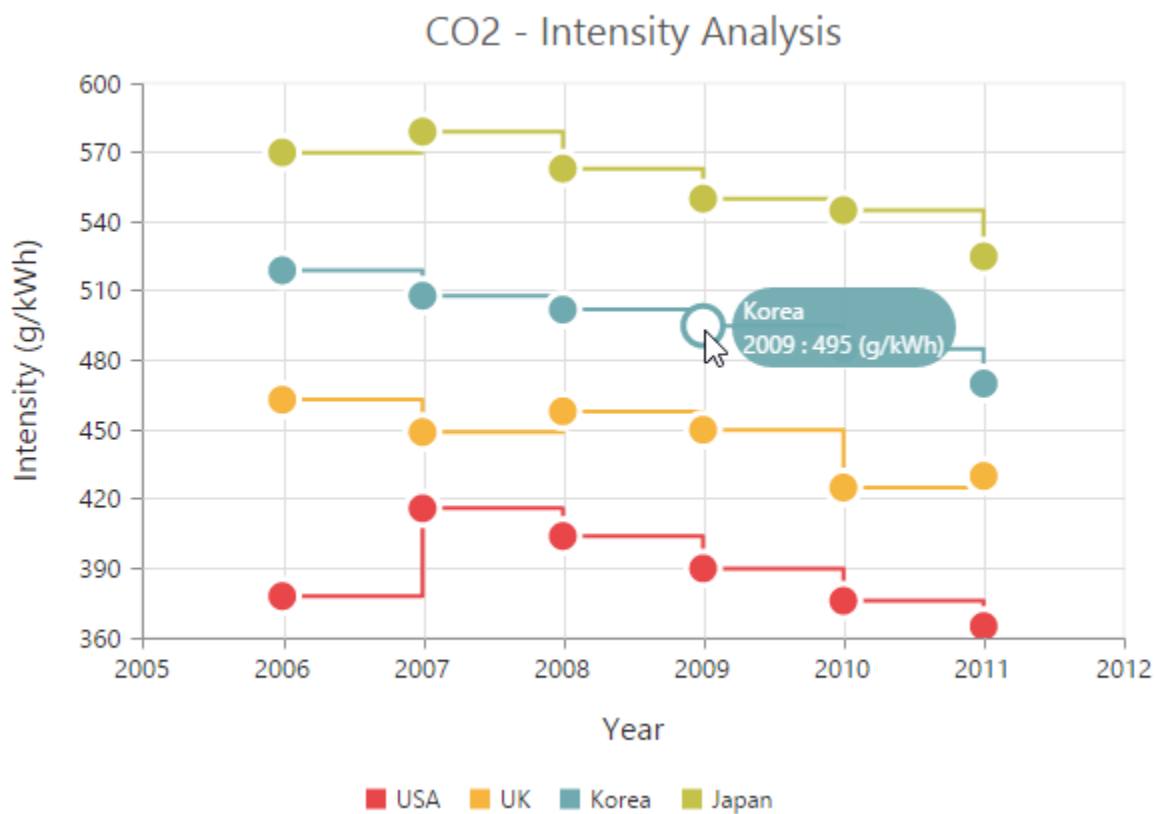


### Tooltip with rounded corners

The options [rx](#) and [ry](#) are used to customize the corner radius of the tooltip rectangle.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series: [{
    tooltip: {
      //Customize the corner radius of the tooltip rectangle.
      rx: "50", ry: "50"
    },
    // ...
  }],
  // ...
});
```



### Zooming and Panning

#### Enable Zooming

There are two ways you can zoom the chart,

- When the [zooming.enable](#) option is set to true, you can zoom the chart by using the rubber band selection.

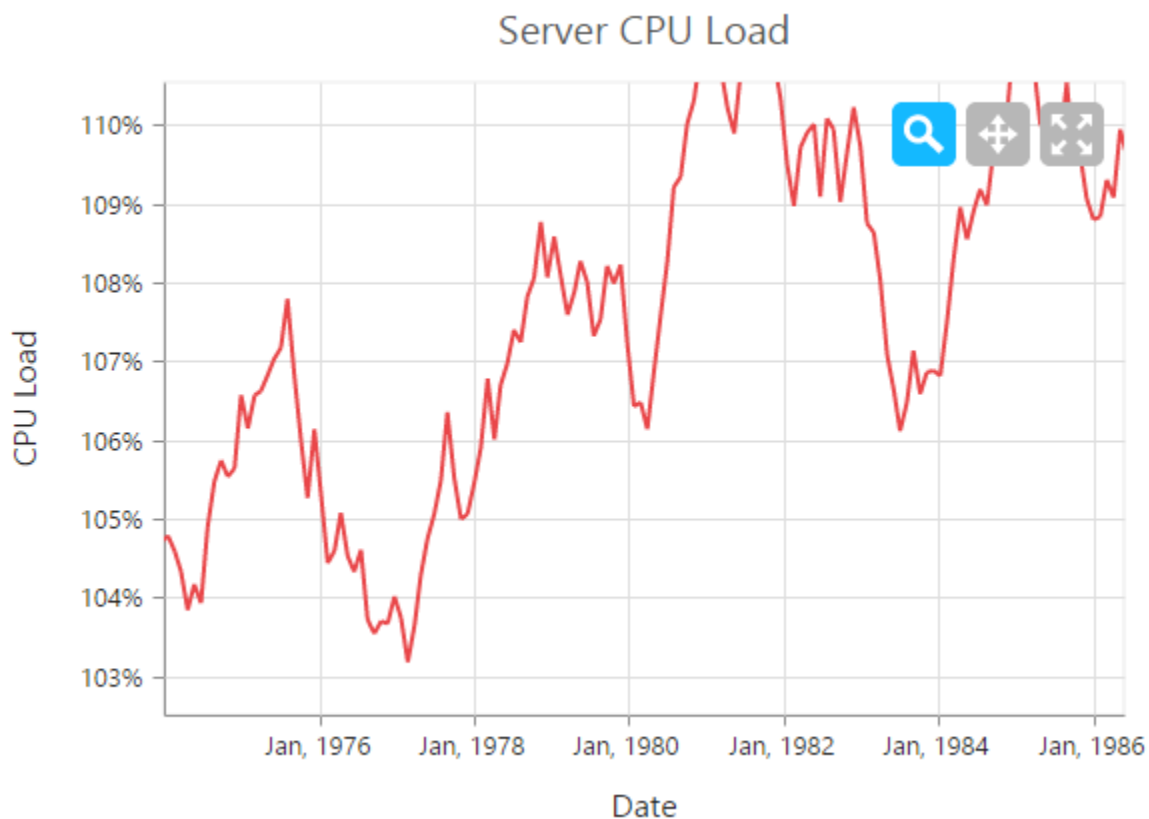


- When the [zooming.enableMouseWheel](#) option is set to true, you can zoom the chart on mouse wheel scrolling.
- When [zooming.enablePinching](#) option is set to *true*, you can zoom the chart through pinch gesture.

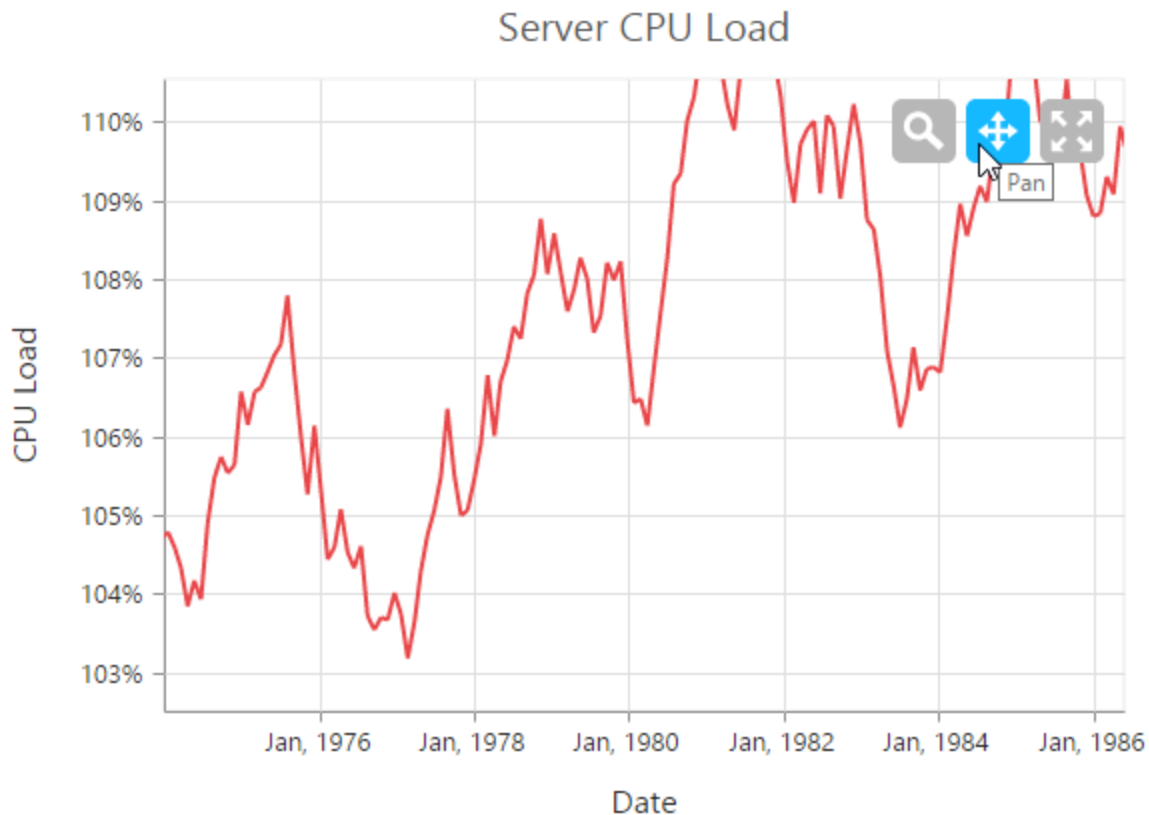
**Note:** Pinch zooming is supported only in browsers that support multi-touch gestures. Currently IE10, IE11, Chrome and Opera browsers support multi-touch in desktop devices.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  //Enable zooming in chart  
  zooming: {enable: true}  
  // ...  
});
```



After zooming the chart, a zooming toolbar will appear with options to *zoom*, *pan* and *reset*. Selecting the Pan option will allow to pan the chart and selecting the Reset option will reset the zoomed chart.



### Types of zooming

The [type](#) option in zooming specifies whether the chart is allowed to scale along the horizontal axis or vertical axis or along both axis. The default value of the [type](#) is “xy” (both axis).

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  zooming: {
    enable: true,
    //Enable horizontal zooming
    type: 'x'
  }
  // ...
});
```

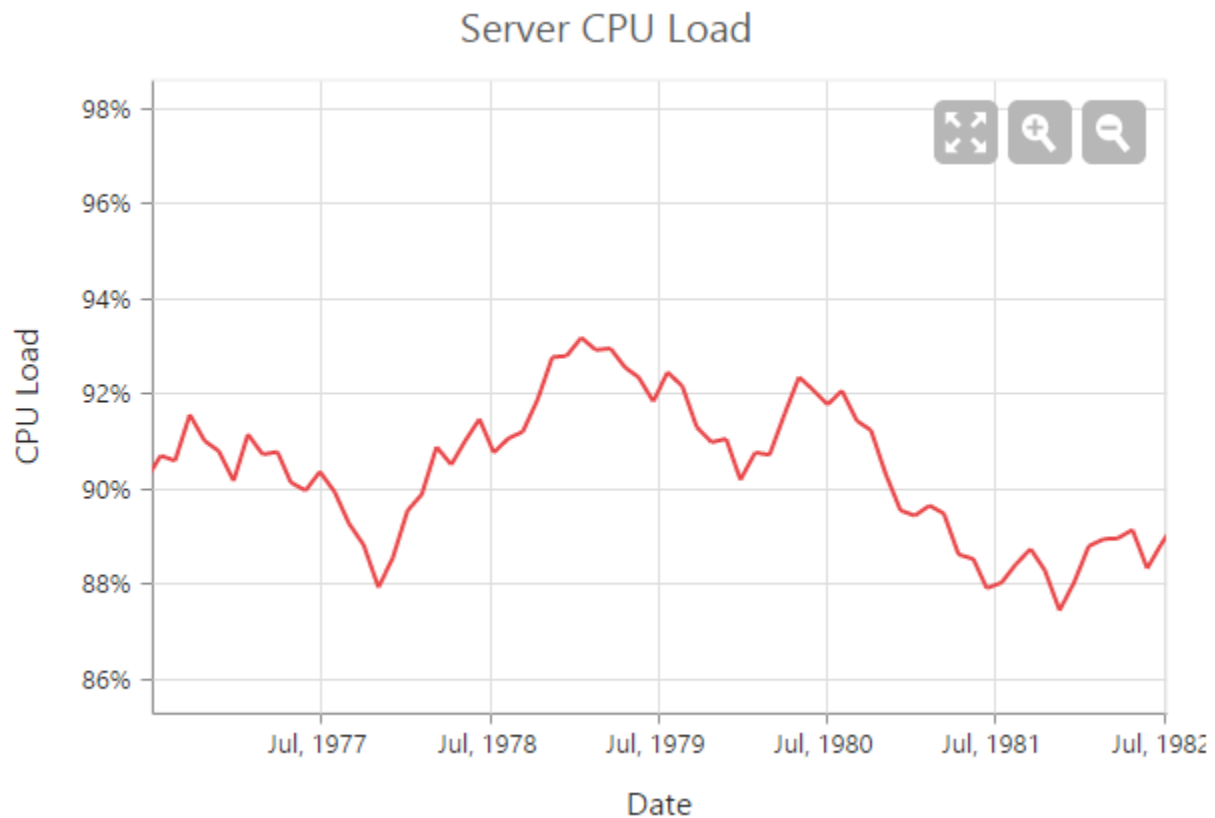
### Customizing zooming toolbar

You can choose the items displayed in the zooming toolbar by specifying the property [toolBarItems](#).

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  //Customizing zooming toolbar
  zooming: {
    enable: true,
    toolbarItems: ["reset", "zoomIn", "zoomOut"]
  }
});
```

```
},
// ...
});
```

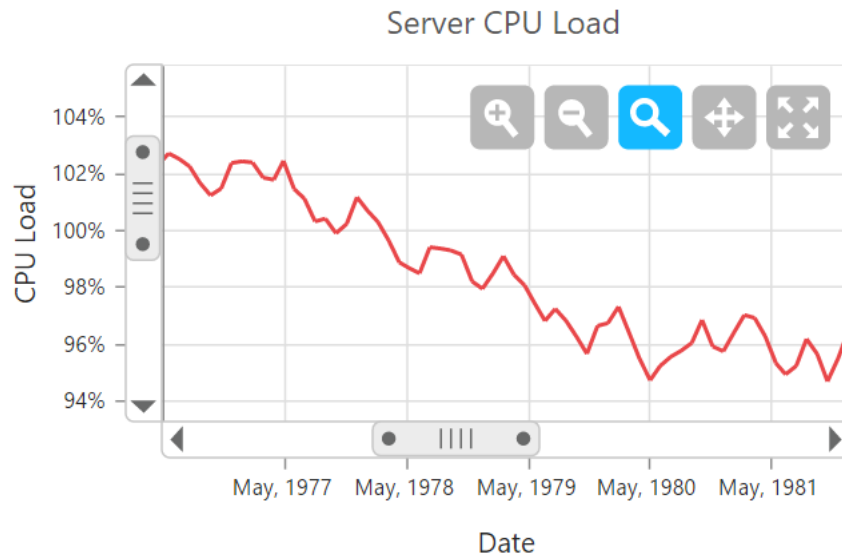


#### Enable ScrollBar

EjChart provides scrollbar support to view the other portions of chart area which is not shown in the view port when zoomed, by setting true to [enableScrollbar](#) option in [zooming](#).

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
//Enable zooming scrollbar in chart
zooming: {enable:true, enableScrollbar: true}
// ...
});
```



### Crosshair

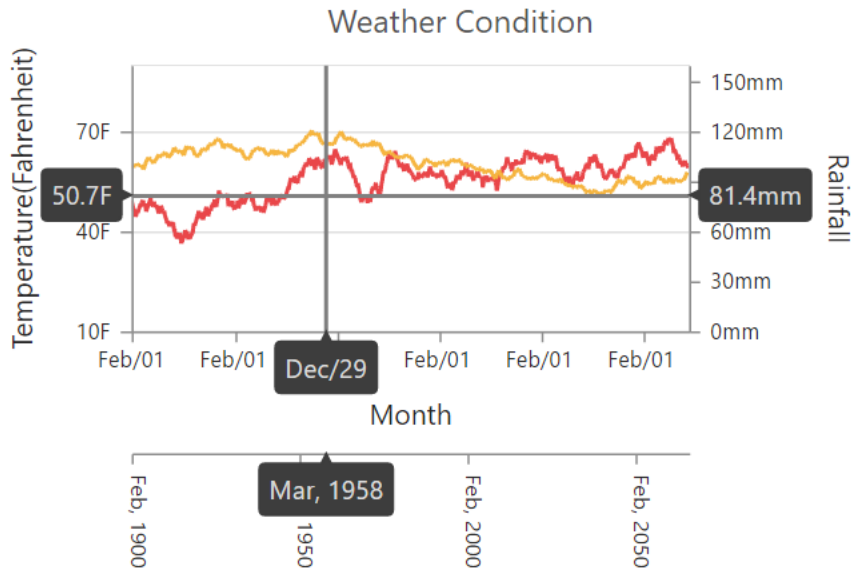
Crosshair is used to view the value of an axis at mouse position or touch contact point.

#### *Enable crosshair and crosshair label*

Crosshair can be enabled by using the [visible](#) option in the [crosshair](#). Crosshair label for an axis can be enabled by using the [visible](#) option of [crosshairLabel](#) in the corresponding axis.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  primaryXAxis:{
    //Enable crosshairLabel to X-Axis
    crosshairLabel: { visible: true },
  },
  primaryYAxis:{
    //Enable crosshairLabel to Y-Axis
    crosshairLabel: { visible: true },
  },
  //Initializing Crosshair
  crosshair:{
    visible: true
  },
  // ...
});
```

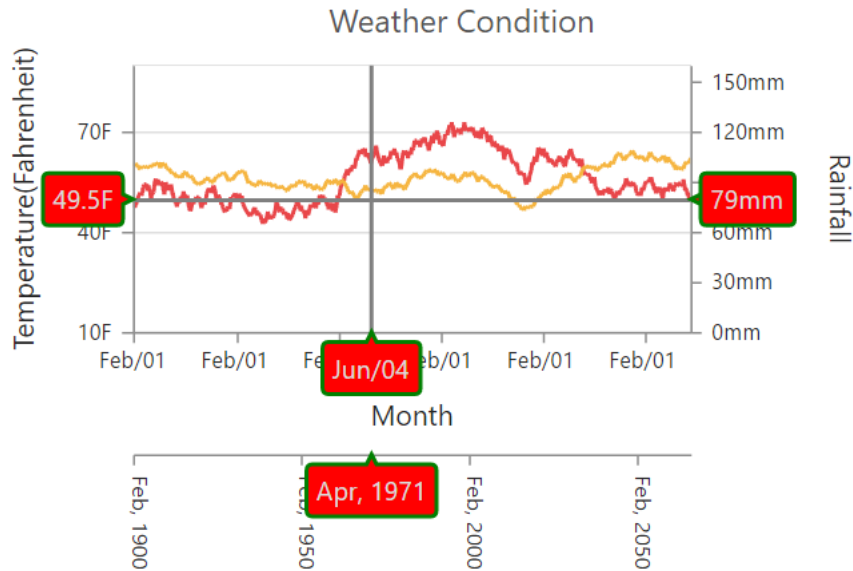


#### Customize the crosshair line and crosshair label

The [fill](#) and [border](#) options of the [crosshairLabel](#) is used to customize the background color and border of the crosshair label respectively. Color and width of the crosshair line can be customized by using the [line](#) option in the [crosshair](#).

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  primaryXAxis: {
    //...
    crosshairLabel: {
      visible: true,
      //Customizing the crosshair label background color and border
      fill: "red",
      border: { color: "green", width: 2 }
    },
    crosshair: {
      visible: true,
      //Customizing the crosshair line
      line: { color: 'gray', width: 2 },
    }
    // ...
  });
```



### Trackball

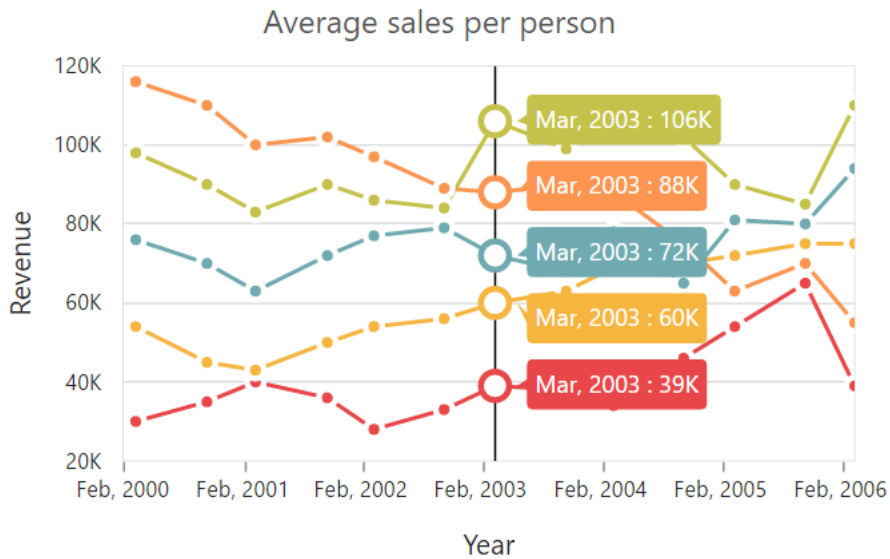
Trackball is used to track a data point close to the mouse position or touch contact point. Trackball marker indicates the closest point and trackball tooltip displays the information about the point.

#### Enable Trackball

Trackball can be enabled by setting the [visible](#) option of the crosshair to *true* and then set the [type](#) as **“trackball”**. The default value of type is **“crosshair”**.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  crosshair: {
    visible: true,
    //Change crosshair type to trackball
    type: 'trackball',
  }
  //.....
});
```

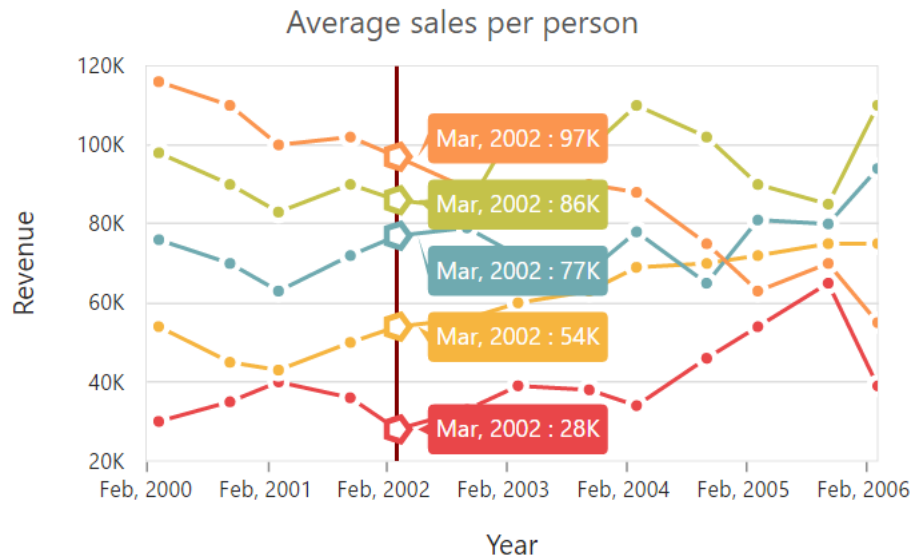


#### Customize trackball marker and trackball line

Shape and size of the trackball marker can be customized by using the [shape](#) and [size](#) options of the crosshair marker. Color and width of the trackball line can be customized by using the [line](#) option in the crosshair.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  crosshair: {
    visible: true,
    //Customize the trackball line color and width
    line: { color: '#800000', width: 2 },
    //Customize the trackball marker shape size and visibility
    marker: {
      shape: 'pentagon',
      size: {
        height: 9, width: 9
      },
    },
    //Enable/disable trackball marker
    visible: true
  }
  // ...
});
```



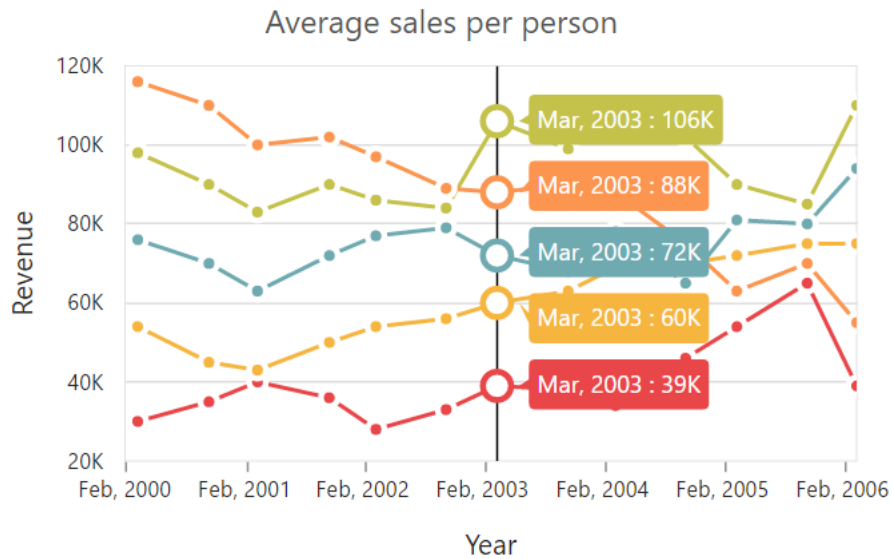
#### Format Trackball tooltip

X and Y values displayed in the trackball tooltip are formatted based on its axis [labelFormat](#).

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  primaryXAxis: {
    labelFormat: 'MMM, yyyy',
    //...
  },
  primaryYAxis: {
    labelFormat: '{value}K',
    //...
  },
  crosshair: {
    visible: true,
    type: 'trackball',
  }
  // ...
});
```



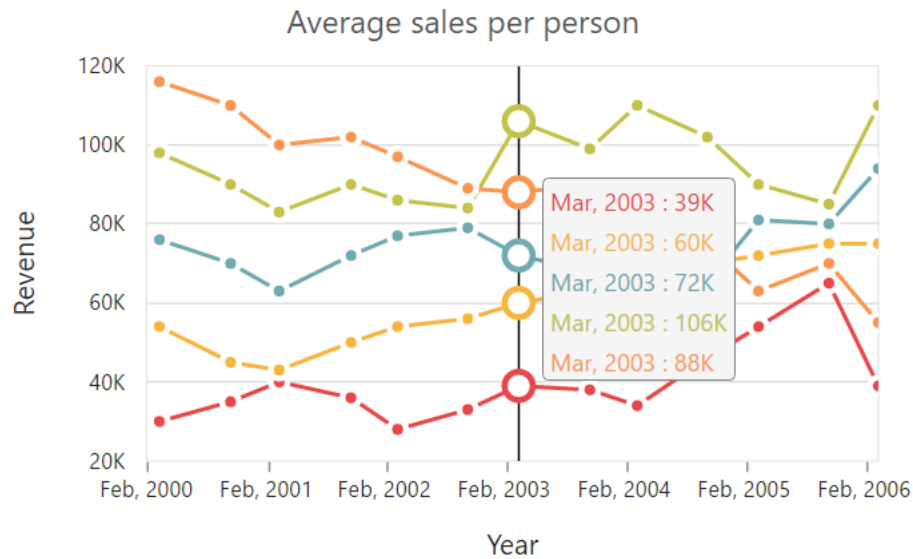


You can able to show the trackball tooltip in two modes, using trackballTooltipSettings.

1. Grouping. 2. Float.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  crosshair: {
    visible: true,
    type: 'trackball',
    //Customize the trackball tooltip
    trackballTooltipSettings: {
      //Trackball mode
      mode: 'grouping',
      //Customize the trackball border, fill, rx and ry
      border:{
        width:1,
        color: 'Grey'
      },
      rx: 3,
      ry: 3,
      fill: 'whitesmoke'
    }
  }
  // ... });
```

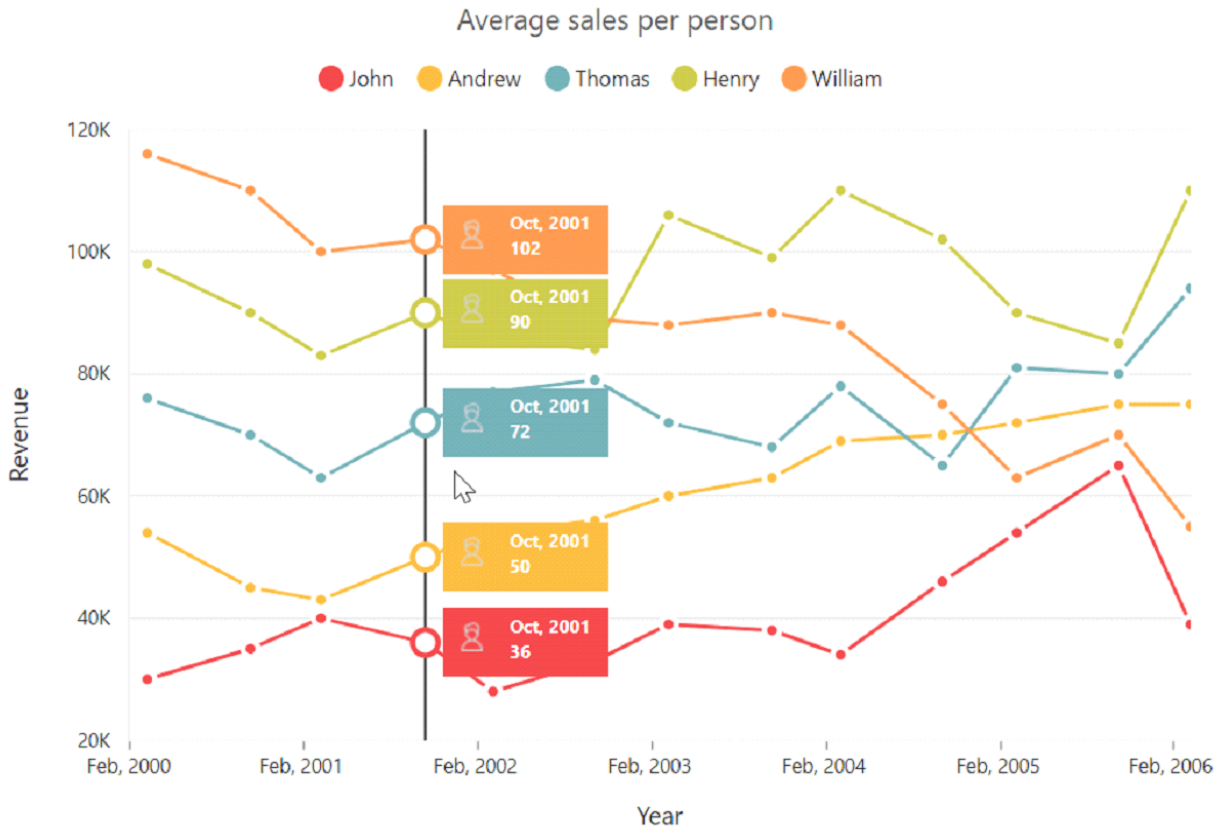


#### Trackball tooltip template:

Trackball tooltip template is used to display the tooltip in customized template format. You can define the desired template in css style. You can enable the **toolTipTemplate** by using the following code snippet.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  crosshair: {
    visible: true,
    type: 'trackball',
    //Customize the trackball tooltip
    trackballTooltipSettings: {
      //Trackball mode
      mode: 'float',
      toolTipTemplate: 'template'
    }
  }
  // ... });
```



### Highlight

EjChart provides highlighting support for the series and data points on mouse hover. To enable the highlighting option, set the [enable](#) property to *true* in the [highlightSettings](#) of the series.

**Note:** When hovering mouse on the data points, the corresponding series legend also will be highlighted.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    highlightSettings: {
      // enable the highlight settings
      enable: true,
    },
    // ...
  }]
  // ...
});
```

### Highlight Mode

You can set three different highlight mode for the highlighting data point and series by using the [mode](#) property of the [highlightSettings](#).

- Series

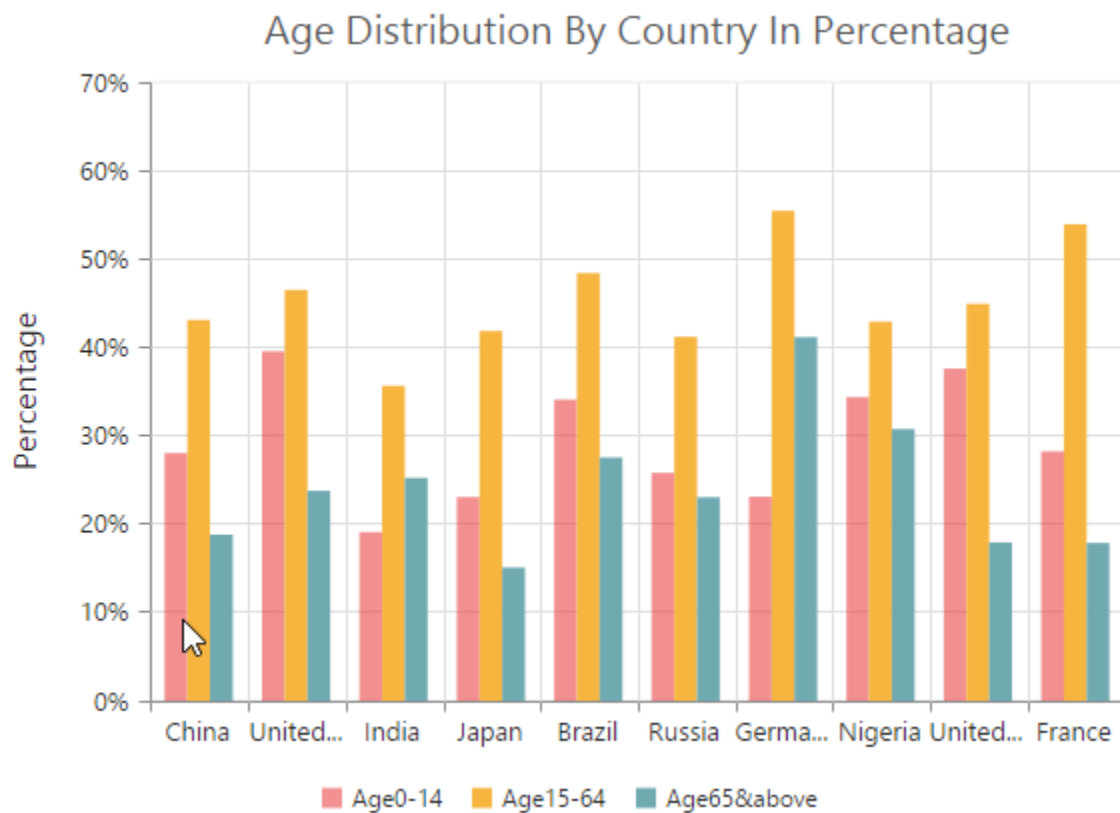
- Points
- Cluster

### Series mode

To highlight all the data points of the specified series, you can set the “**series**” value to the [mode](#) option in the highlightSettings.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    highlightSettings: {
      enable: true,
      //Change highlight mode
      mode: 'series'
    },
    // ...
  }]
  // ...
});
```

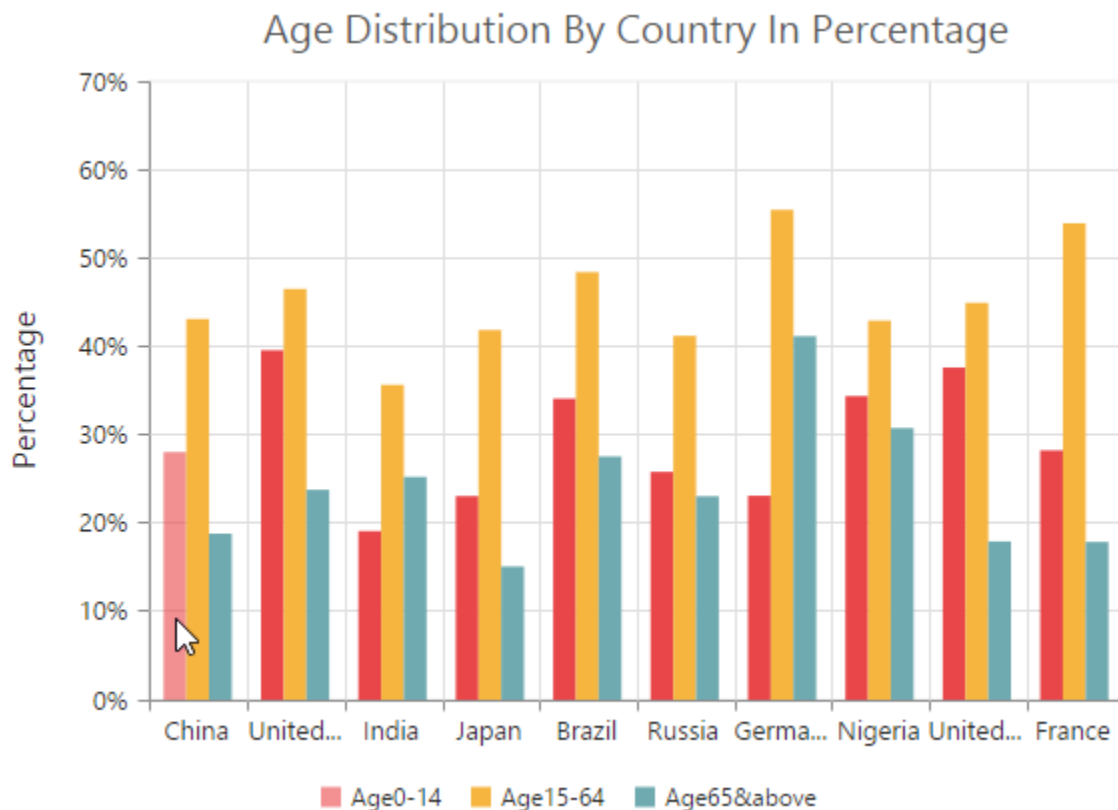


### Point mode

To highlight a single point, you can set the “**point**” value to the [mode](#) option.

**JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    highlightSettings: {
      enable: true,
      //Change highlight mode
      mode: 'point'
    },
    // ...
  }]
  // ...
});
```

**Cluster mode**

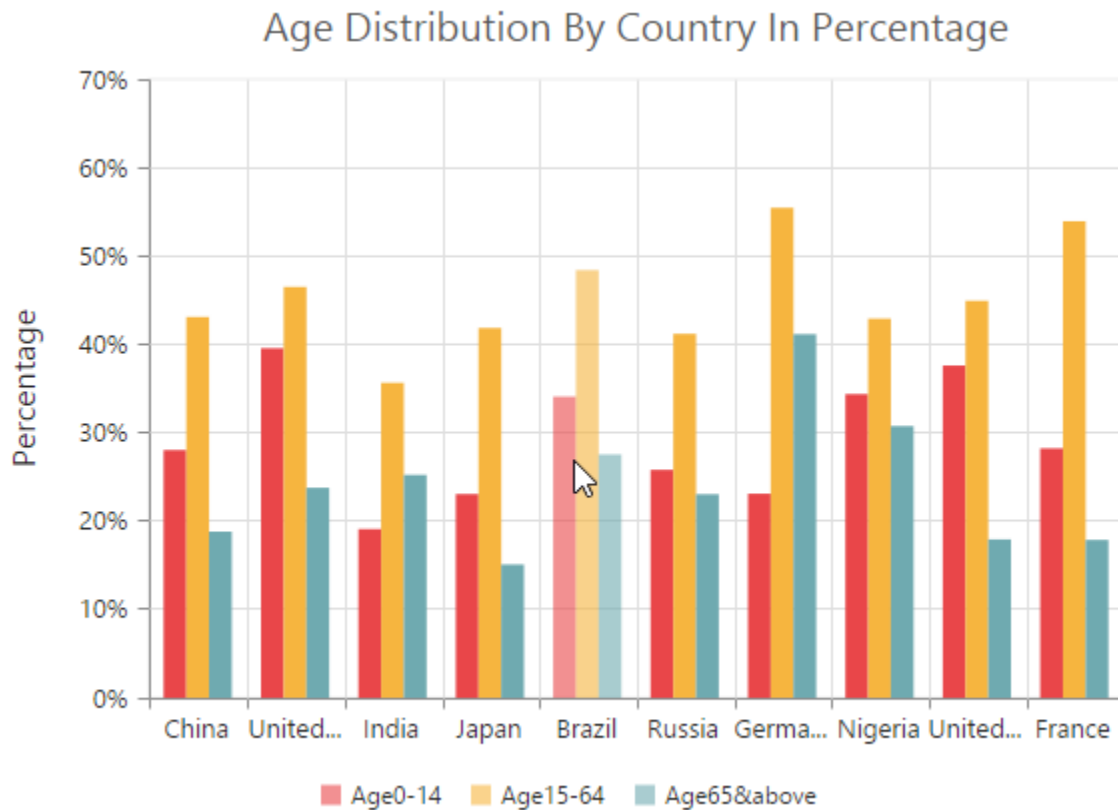
To highlight the points that corresponds to the same index in all the series, set the “**cluster**” value to the [mode](#) option.

**JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    highlightSettings: {
      enable: true,

```

```
//Change highlight mode
mode: 'cluster'
},
// ...
}]
// ...
});
```

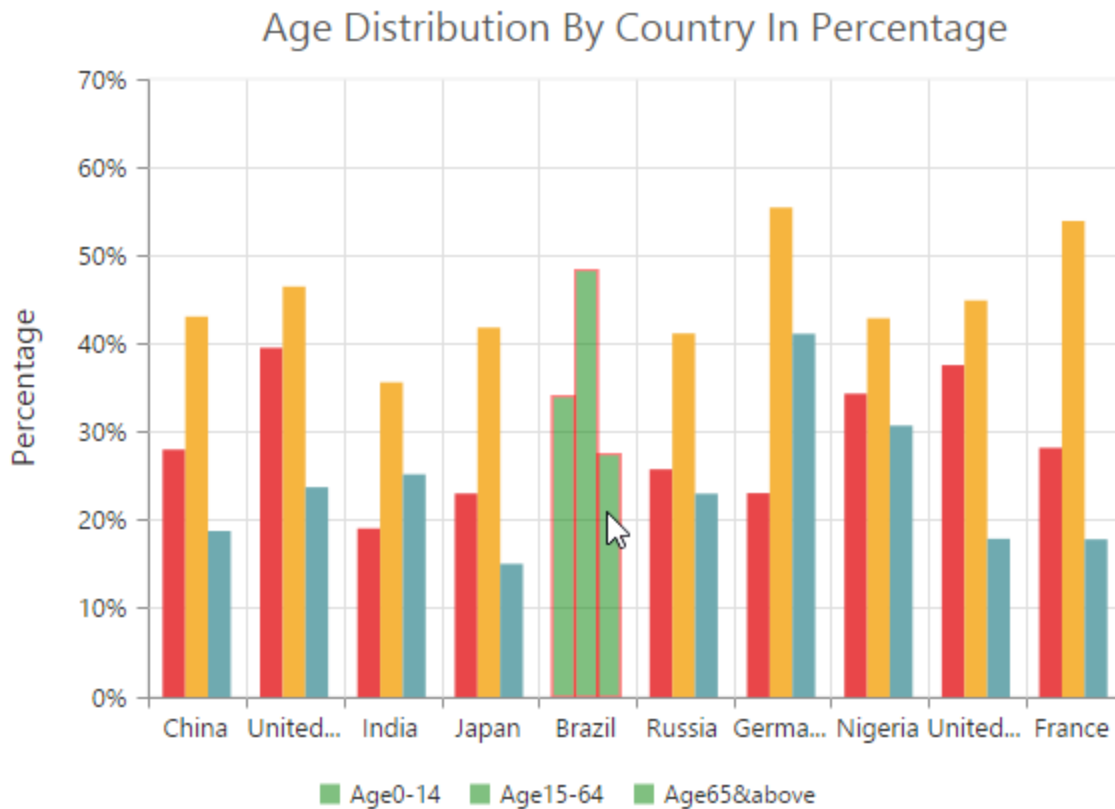


#### Customize the highlight styles

To customize the highlighted series, use the [color](#), [border](#) and [opacity](#) options in the highlightSettings.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
series:[{
highlightSettings: {
enable: true,
//Customizing
border: { width: '1.5', color: "red" },
opacity: 0.5, color: "green"
},
// ...
}]
// ...
});
```



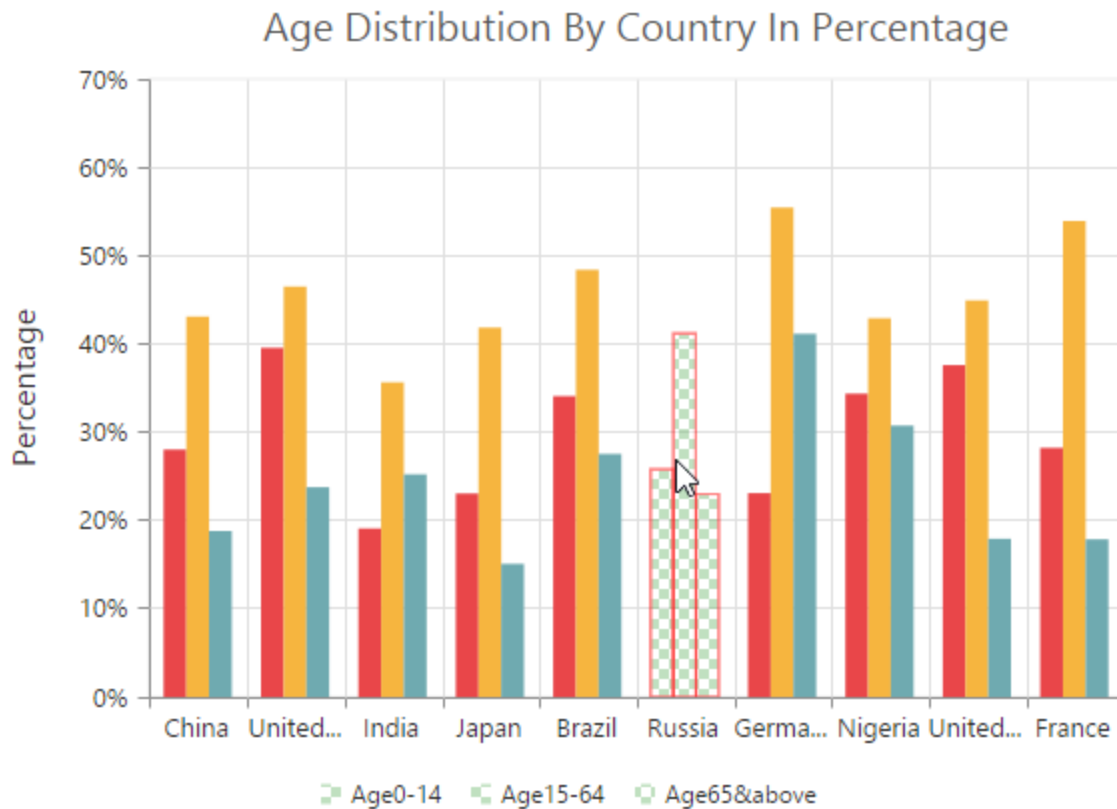
#### Patterns to highlight

EjChart provides pattern support for highlighting the data by setting the value to the [pattern](#) property of the highlightSettings. The different types of highlight patterns are as follows.

1. chessboard 2. crosshatch 3. dots 4. pacman 5. grid 6. turquoise 7. star 8. triangle 9. circle 10. tile 11. horizontalDash 12. verticalDash 13. rectangle 14. box 15. verticalStripe 16. horizontalStripe 17. bubble 18. diagonalBackward 19. diagonalForward

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    highlightSettings: {
      enable: true,
      //Change highlighting pattern
      pattern: "chessboard",
    },
    // ...
  }]
  // ...
});
```



#### Custom pattern

To create a custom pattern for the highlighting data points, set the pattern type as **“custom”** and add the custom pattern id in the [customPattern](#) option of the highlightSettings.

#### HTML

```
<body>
<div id="container"></div>
<svg>
<pattern id="dots_a" patternUnits="userSpaceOnUse" width="6" height="6">
<rect x="0" y="0" width="6" height="6" transform="translate(0,0)"
fill="black" opacity="1"></rect>
<path d='M 3 -3 L -3 3 M 0 6 L 6 0 M 9 3 L 3 9' stroke-width="1"
stroke="white"></path>
</pattern>
</svg>
<script type="text/javascript">
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
series:[{
highlightSettings: [{
enable: true,
//Add custom pattern for highlighting data
pattern: "custom",
customPattern: 'dots_a',
// ...
}],
}],
}],
```



```
// ...  
});  
</script>  
</body>
```

### Selection

EjChart provides selection support for the series and data points on mouse click. To enable the selection option, set the [enable](#) property to *true* in the [selectionSettings](#) of the series.

**Note:** When mouse is clicked on the data points, the corresponding series legend also will be selected.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series:[{  
    selectionSettings: {  
      // enable the selection settings  
      enable: true,  
    },  
    // ...  
  }]  
  // ...  
});
```

### Selection Mode

You can set four different selection mode for highlighting the data point and series by using the [mode](#) property of the selectionSettings.

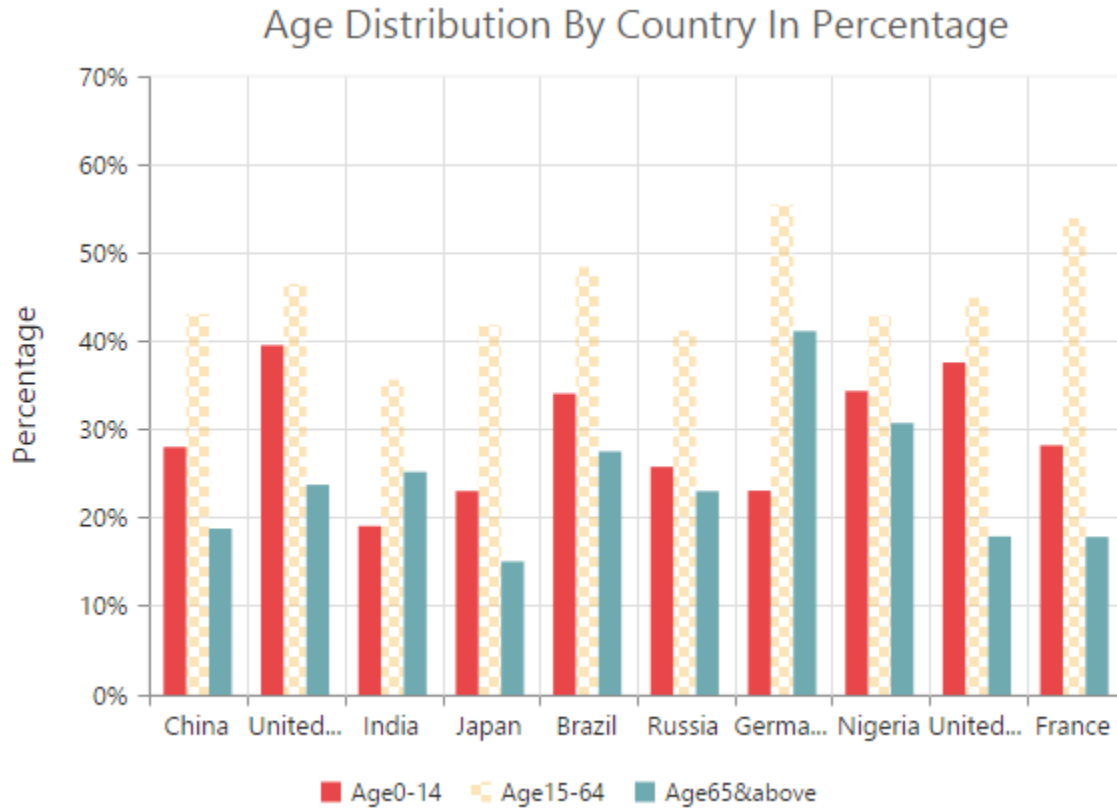
- Series
- Points
- Cluster
- Range

### Series mode

To select all the data points of the specified series, you can set the “series” value to the [mode](#) option in the selectionSettings.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  series:[{  
    selectionSettings: {  
      enable: true,  
      //Change selection mode  
      mode: 'series',  
      pattern: 'chessboard'  
    },  
    // ...  
  }]  
  // ...  
});
```

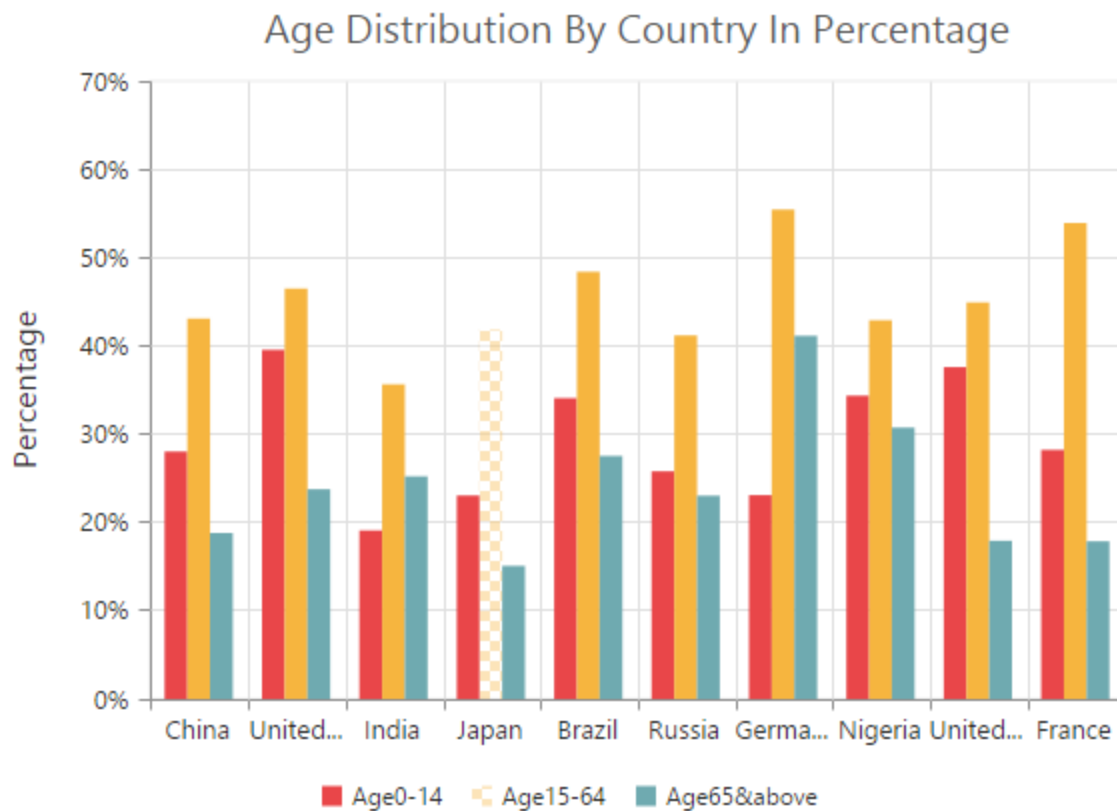


#### Point mode

To highlight a single point, you can set the “**point**” value to the [mode](#) option.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    selectionSettings: {
      enable: true,
      //Change selection mode
      mode: 'point',
      // ...
    },
    // ...
  }]
  // ...
});
```

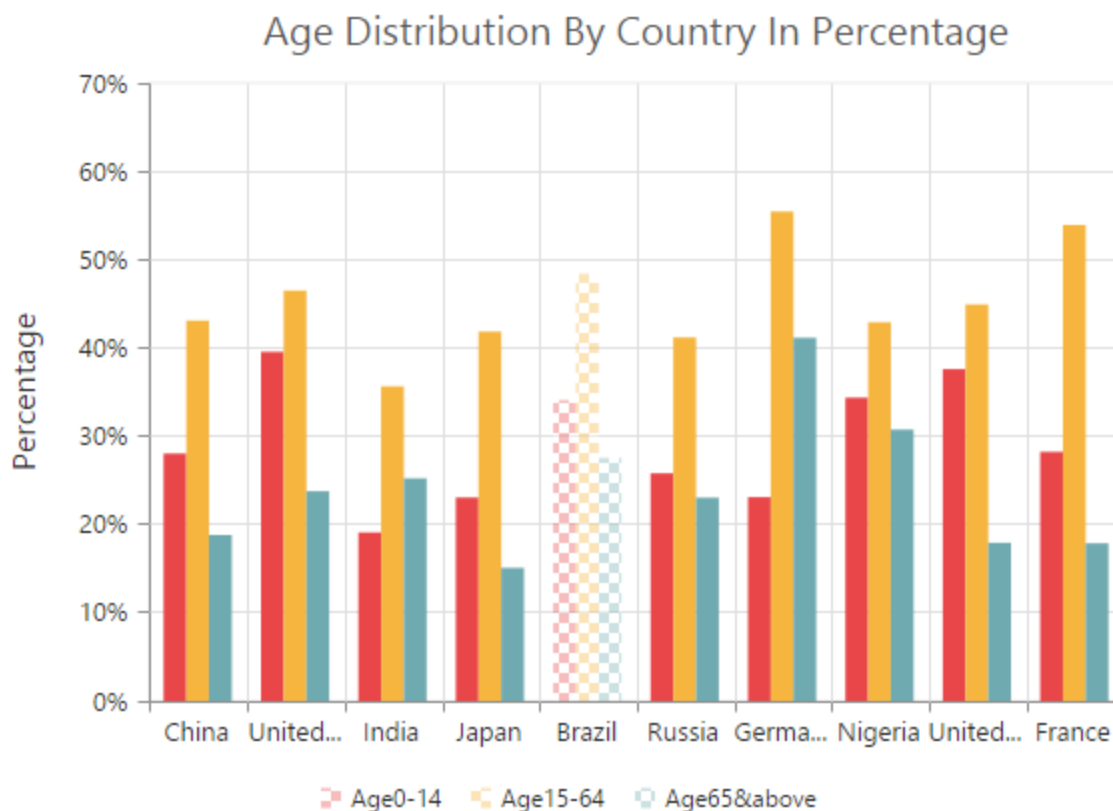


#### Cluster mode

To select the points that corresponds to the same index in all the series, set the **“cluster”** value to the [mode](#) option.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    selectionSettings: {
      enable: true,
      //Change selection mode
      mode: 'cluster',
      // ...
    },
    // ...
  }]
  // ...
});
```



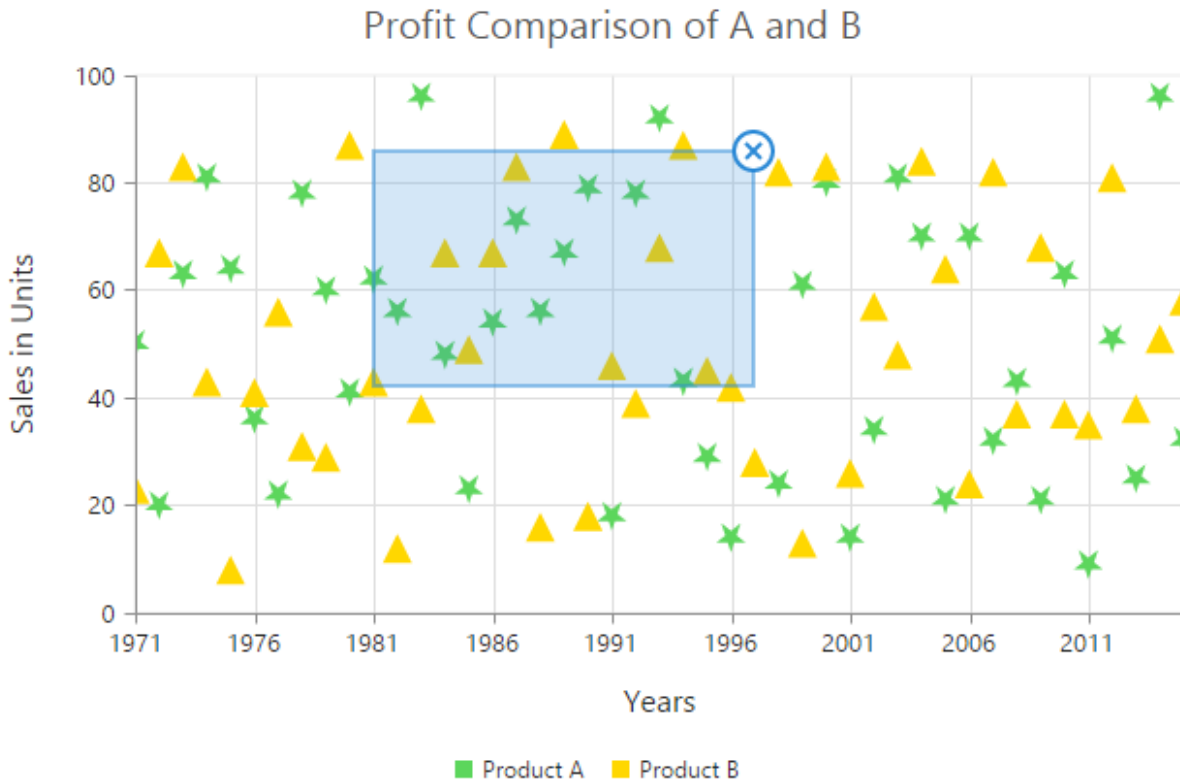
### Range mode

To fetch the selected area data points value, you can set the selectionSettings [mode](#) as **range** in the chart series. The selection rectangle can be drawn as horizontally, vertically or in both direction by using [rangeType](#) property and the selected data's are returned as an array collection in the [rangeSelected](#) event.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    selectionSettings: {
      enable: true,
      //Change selection mode
      mode: 'range',
      //Enable both axis selection
      rangeType: 'xy'
      // ...
    },
    // ...
  }],
  //Subscribe the rangeSelected event.
  rangeSelected: "rangeSelection"
  // ...
});
//event to fetch the selected data point values
```

```
rangeSelection:function (sender){
var selectedData = sender.data.selectedDataCollection;
//...
}
```



#### Selection Type

You can set two different selection type for selecting the data point and series on mouse click by using the [type](#) property of the selectionSettings.

- Single
- Multiple

#### Single Type

To select a data point or a series on mouse click based on the [selectionSettings.mode](#), set [selectionSettings.type](#) as “single” in the series.

#### JAVASCRIPT

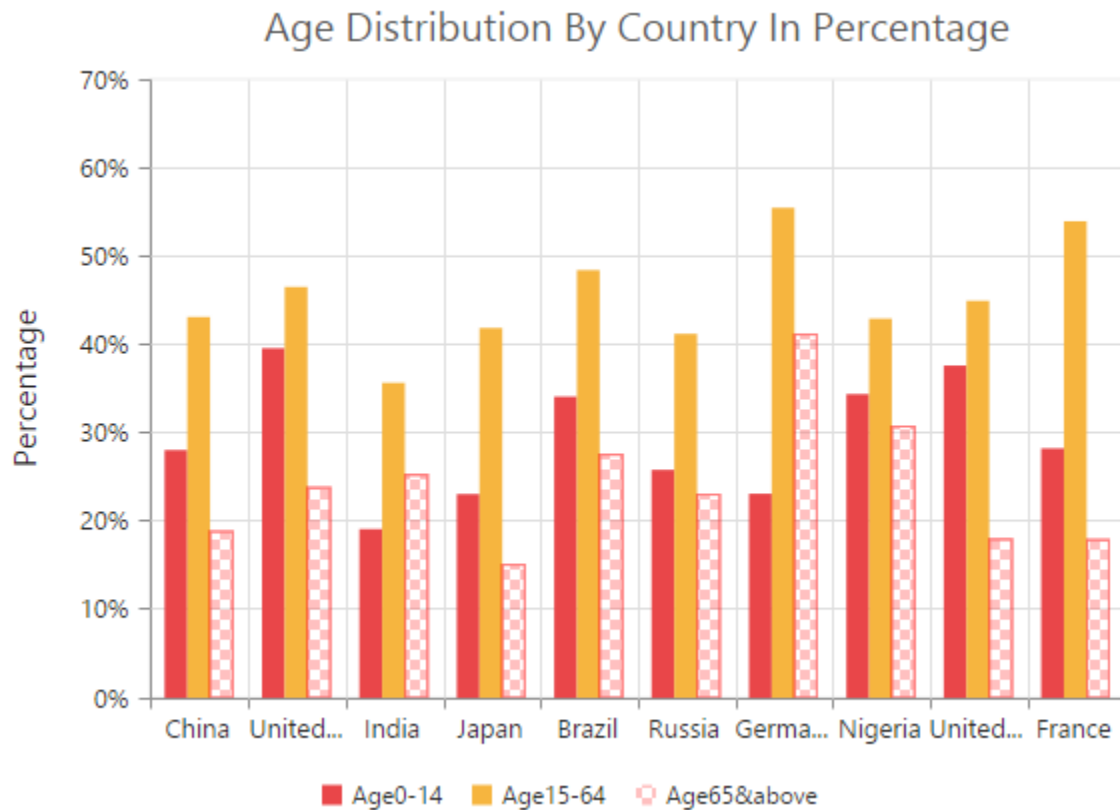
```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
commonSeriesOptions:{
selectionSettings: {
enable: true,
//Selection mode is series or point ,cluster
mode: 'series',
//Selection type is single

```

```

type: 'single'
},
}
// ...
});

```



### Multiple Type

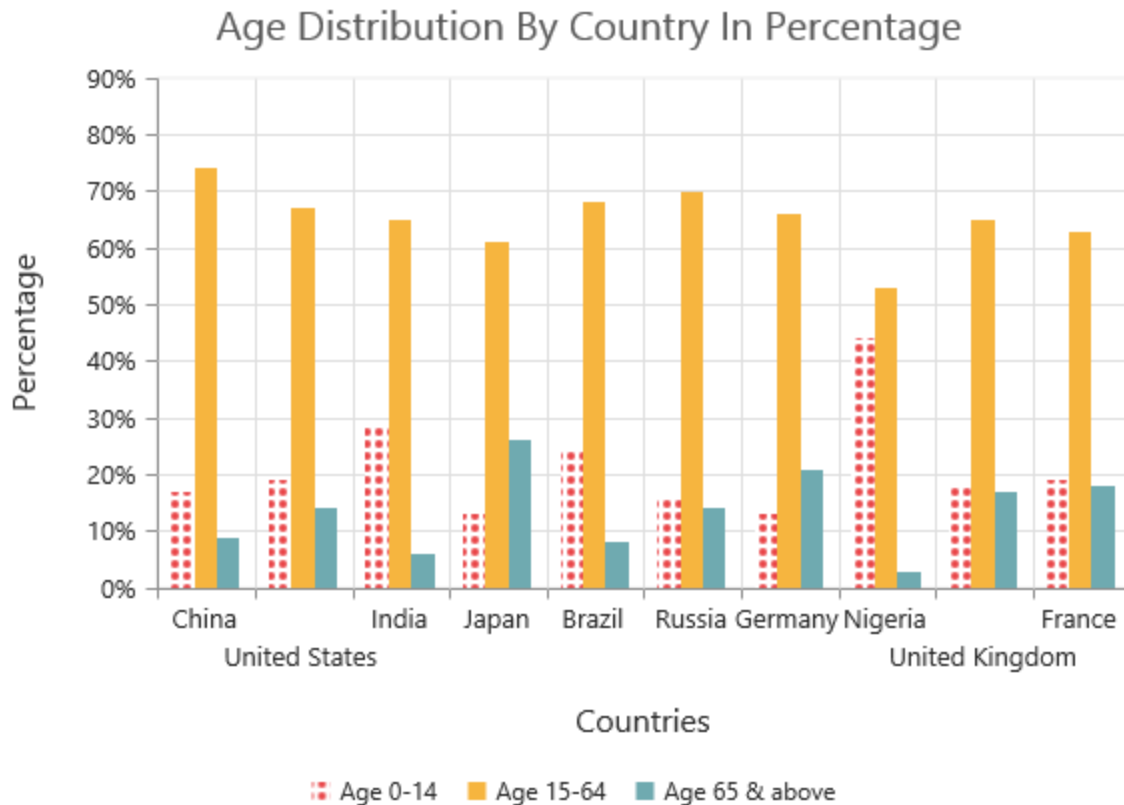
For selecting multiple data points or series on mouse click, set [selectionSettings.type](#) as “multiple” in the series.

### JAVASCRIPT

```

var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
commonSeriesOptions:{
selectionSettings: {
enable: true,
//Selection mode is series or point ,cluster
mode: 'series',
//Selection type is multiple
type: 'multiple'
},
},
// ...
});

```

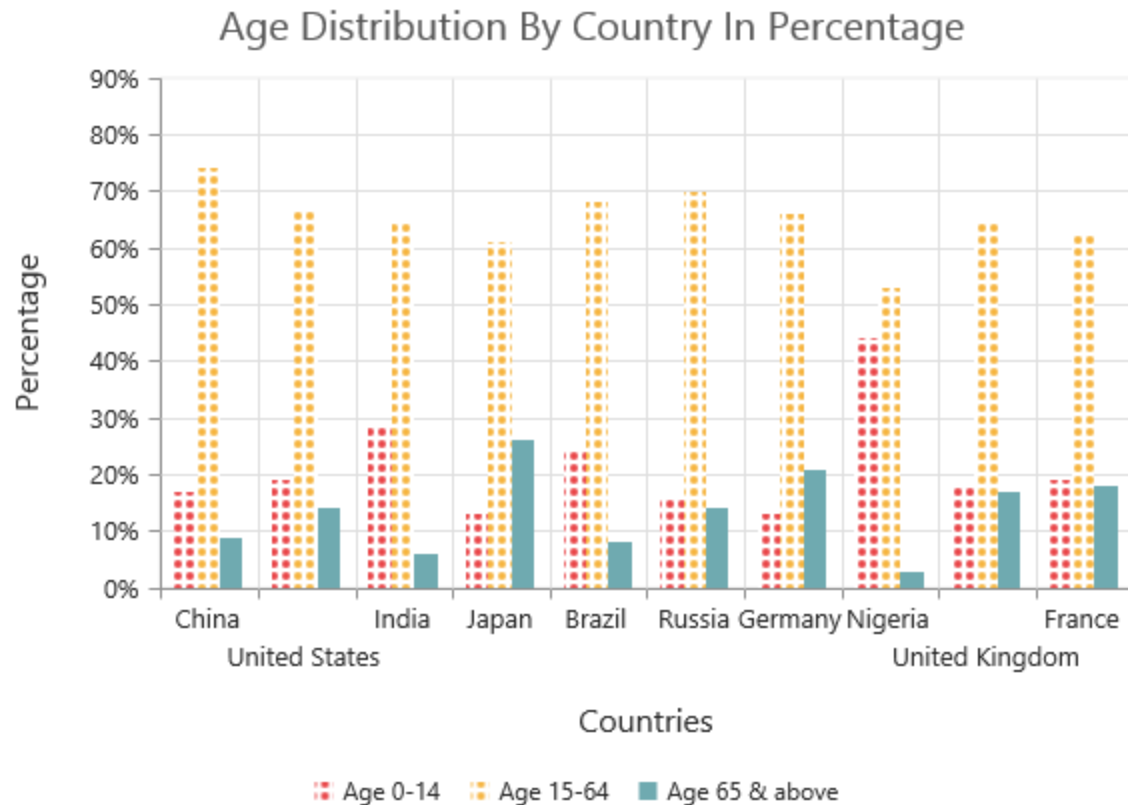


#### Customizing selection styles

To customize the selection styles, use the [color](#), [border](#) and [opacity](#) options in the selectionSettings.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    selectionSettings: {
      enable: true,
      //Customizing selection rectangle styles
      border: { width: '1.5', color: "red" },
      opacity: 0.5, color: "red"
    },
    // ...
  },
  // ...
}]
// ...
});
```



#### Patterns for selection

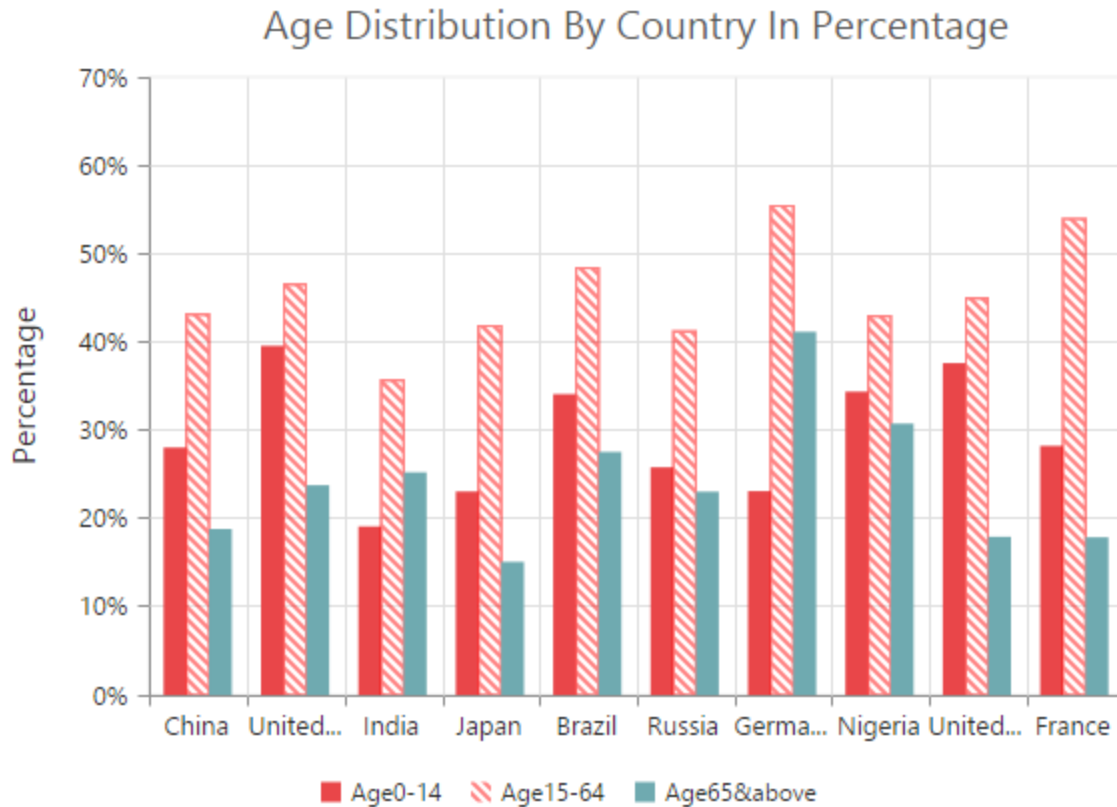
EjChart provides pattern support for the data selection by setting the value to the [pattern](#) property of the selectionSettings. The different types of selection patterns are as follows.

1. chessboard 2. crosshatch 3. dots 4. pacman 5. grid 6. turquoise 7. star 8. triangle 9. circle 10. tile 11. horizontalDash 12. verticalDash 13. rectangle 14. box 15. verticalStripe 16. horizontalStripe 17. bubble 18. diagonalBackward 19. diagonalForward

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    selectionSettings: {
      enable: true,
      //Change selection pattern
      pattern: "diagonalForward",
      // ...
    },
    // ...
  }]
  // ...
});
```





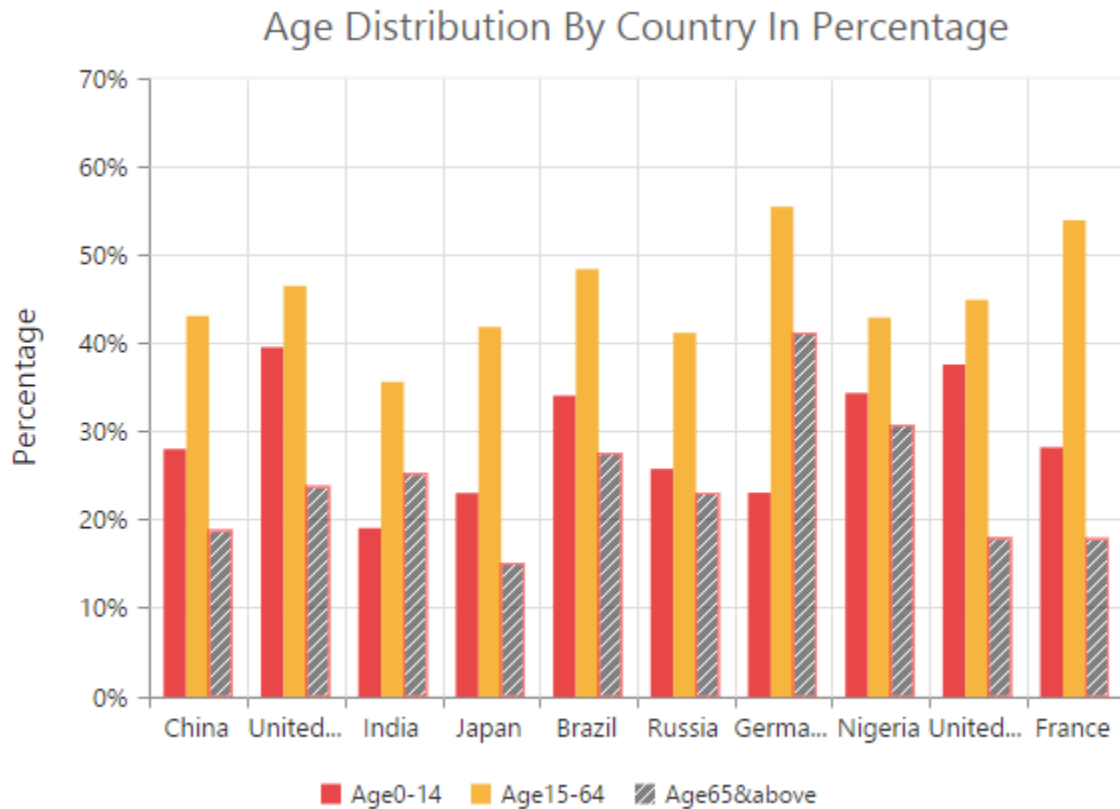
#### Custom pattern

To create a custom pattern for selecting the data points, set the [pattern](#) type as “**custom**” and add the custom pattern [id](#) in the [customPattern](#) option of the selectionSettings.

#### HTML

```
<body>
<div id="container"></div>
<svg>
<pattern id="dots_a" patternUnits="userSpaceOnUse" width="6" height="6">
<rect x="0" y="0" width="6" height="6" transform="translate(0,0)"
fill="black" opacity="1"></rect>
<path d='M 3 -3 L -3 3 M 0 6 L 6 0 M 9 3 L 3 9' stroke-width="1"
stroke="white"></path>
</pattern>
</svg>
<script type="text/javascript">
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
series:[{
selectionSettings: [{
enable: true,
//Add custom pattern for selection data
pattern: "custom",
customPattern: 'dots_a',
// ...
}],
// ...
}
```

```
});
</script>
</body>
```



#### Handling Series Selection

To get the series information when selecting the specific series, subscribe to the [seriesRegionClick](#) event and set the **selectionSettings.mode** as "series".

#### JAVASCRIPT

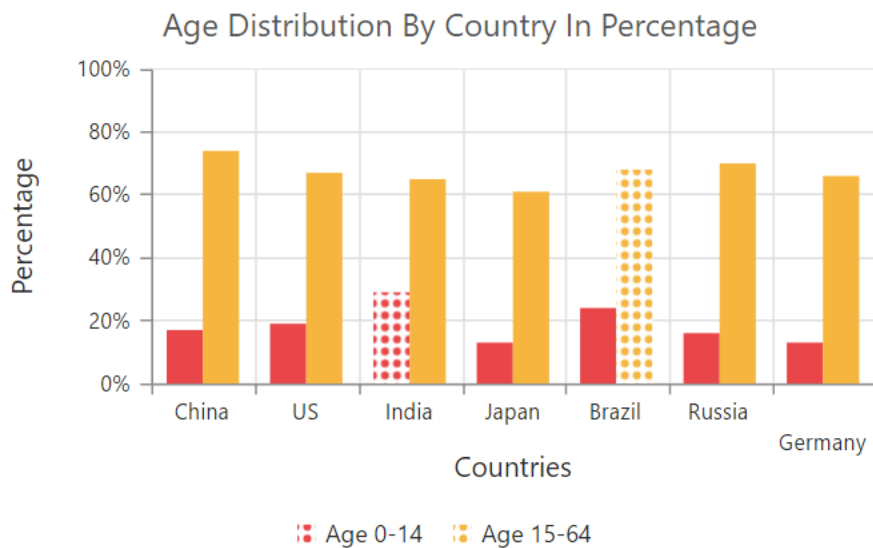
```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series:[{
    selectionSettings: [{
      enable: true,
      //Change selection mode
      mode: "series",
      // ...
    }],
    //Subscribe series selection event
    seriesRegionClick: "seriesSelection",
    // ...
  }];
});
function seriesSelection(sender) {
  //Get Series information on series selection
  var seriesData = sender.series;
}
```

### Selection on Load

We can able to select the point/series programmatically on chart load, by setting series and point index in the `selectedDataPointIndexes` property.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  //Added selected data point indexes
  selectedDataPointIndexes: [
    { seriesIndex:0 , pointIndex:2 },
    { seriesIndex:1 , pointIndex:4 }
  ],
  // ...
});
```

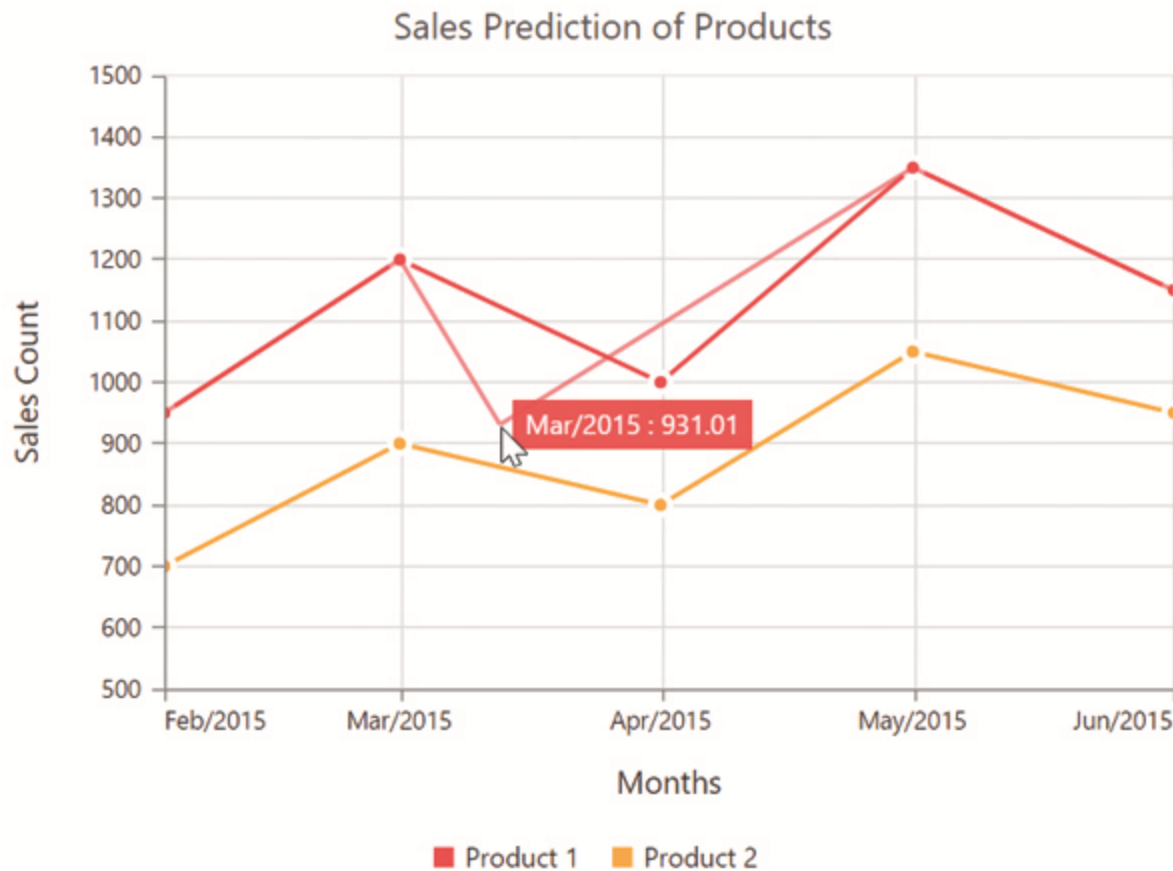


### Data Editing

EjChart provides support to change the location of the rendered points. This can be done by dragging the point and dropping it on another location in chart. To enable the data editing, set the [enable](#) property to true in the [dragSettings](#) of the series.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  //Initializing Series
  series:[{
    dragSettings:{
      enable: true
    }
  }]
});
```

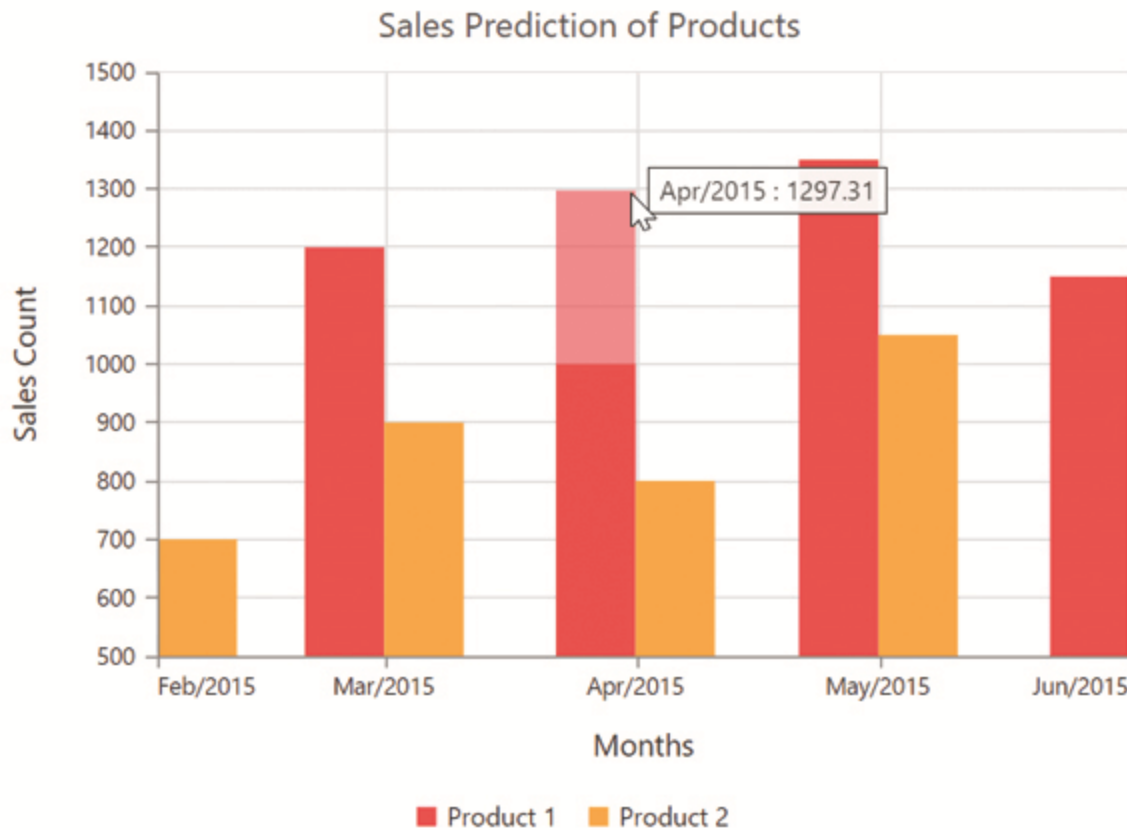


#### Customize Dragging direction

To drag the point along x and y axes, you can specify [type](#) as xy in dragSettings. And to drag along x axis alone, specify the type as x and to drag along y axis, specify type as y.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  {
    //Initializing Series
    series:[{
      dragSettings:{
        type: 'y'
      }
    }]
  });
```



## Performance

- When there are large number of points to load, you can enable canvas rendering mode in chart. Canvas rendering is faster than SVG because it does not involve manipulating DOM elements as much as SVG rendering.

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
  $(function () {
    var chartsample = new ej.datavisualization.Chart($("#container"), {
      // ...
      //Enable Canvas rendering mode
      enableCanvasRendering: true,
      // ...
    });
  });
}

```

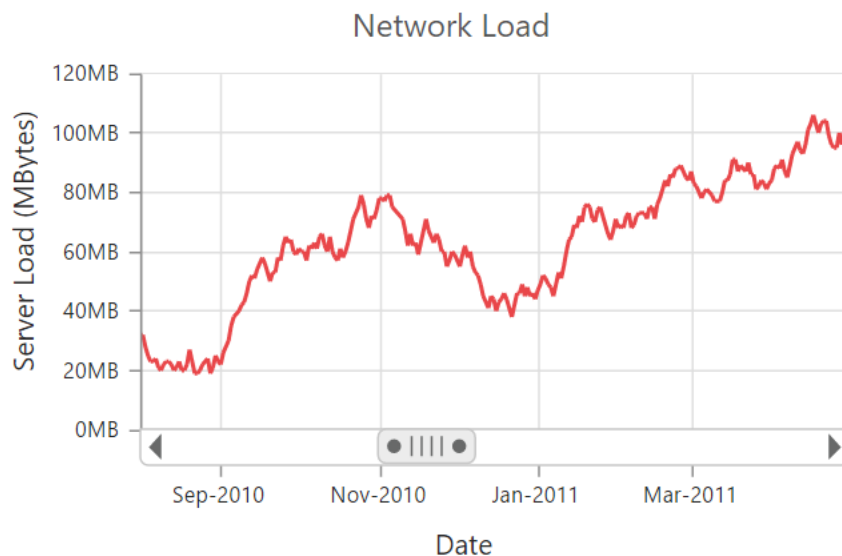
- Instead of enabling data markers and labels when there are large number of data points, you can use **trackball** and **tooltip** to view point information.

### Lazy Loading

Lazy loading feature provides an effective way for loading data on demand by scrolling and viewing a smaller range of data from a larger collection.

### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#container"), {
  primaryXAxis:
  {
    scrollbarSettings: {
      // enable the scrollbar
      visible: true,
      // enable the resize option
      canResize: true,
      range: {
        min: "2009/1/1",
        max: "2014/1/1"
      }
    }
  },
  // . . .
});
```



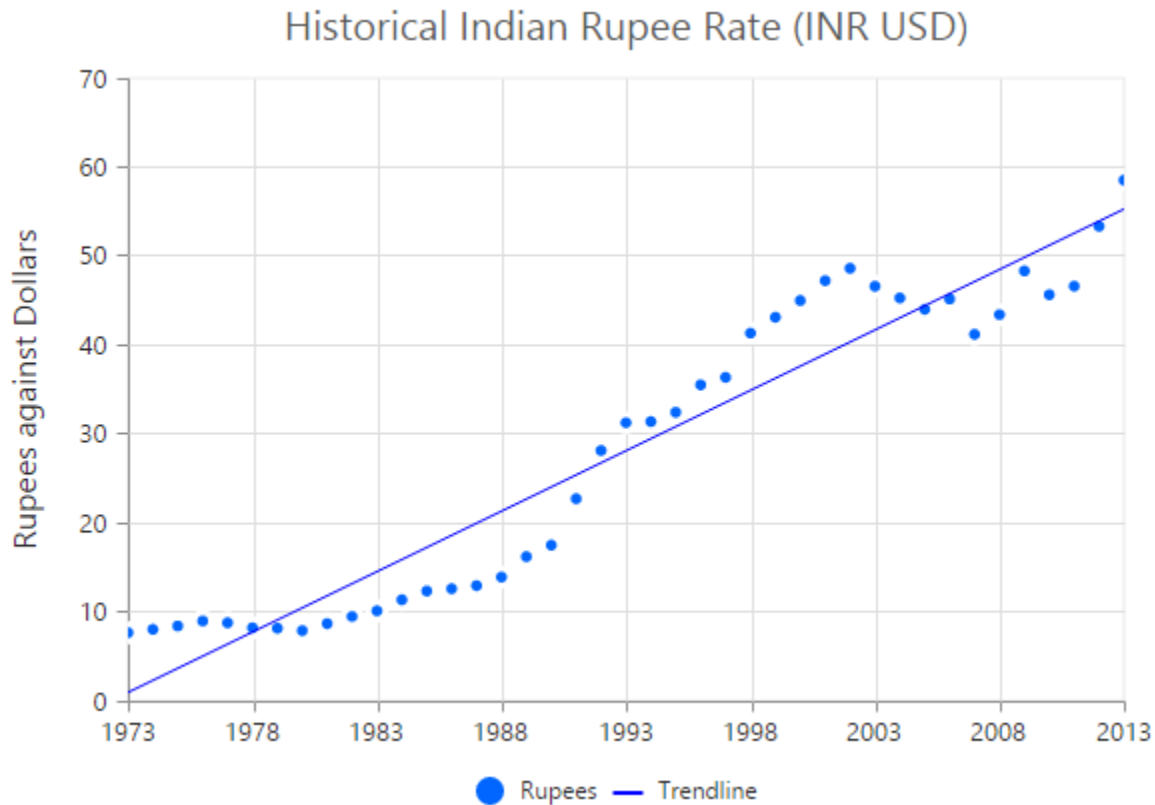
### Trendlines

EjChart can generate Trendlines for Cartesian type series (*Line, Column, Scatter, Area, Candle, HiLo etc.*) except bar type series. You can add more than one trendline object to the [trendlines](#) option.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
  $(function () {
    var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
      series:[{
        trendlines: [{
```

```
//Enable Trendline to chart series
visibility: "visible", type: "linear"
}],
//...
}]
//...
});
});
}
```

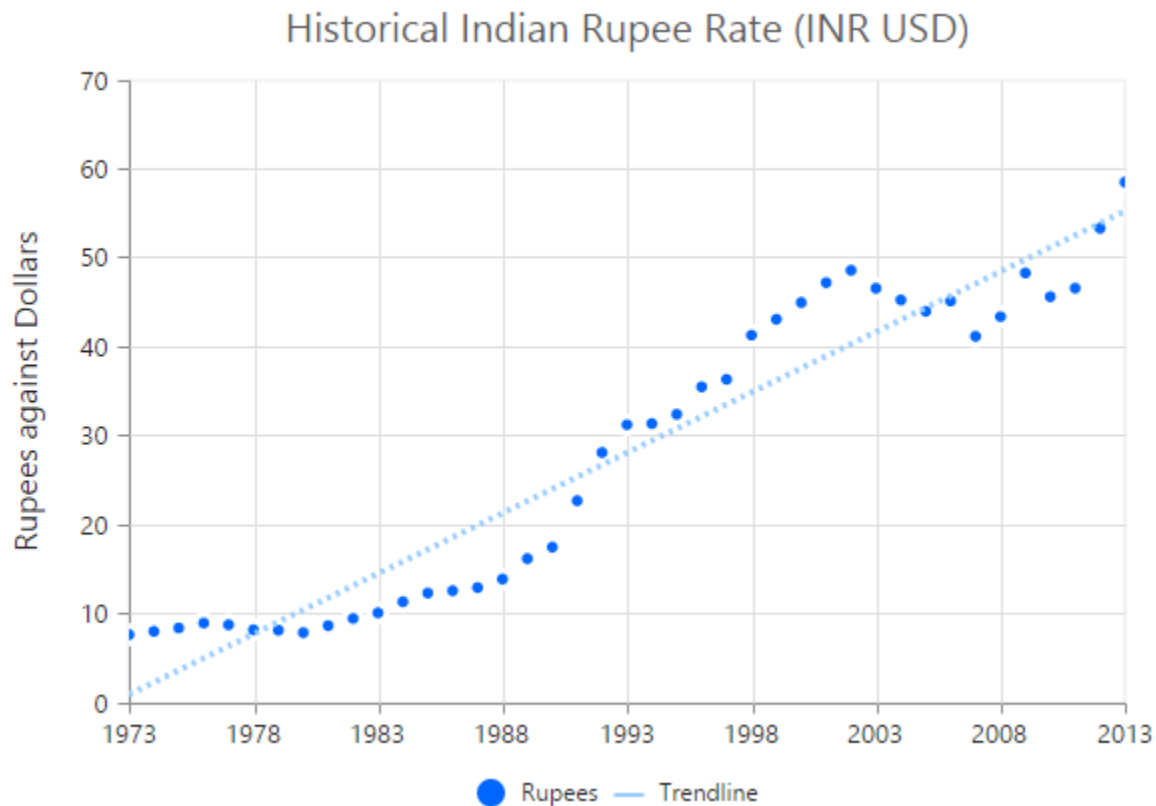


### Customize the trendline styles

A trendline can be customized by using the properties such as [fill](#), [width](#), [dashArray](#) and [opacity](#). The default type of trendline is “linear”.

### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  series:[{
    trendlines: [{
      //...
      //Customize the Trendline styles
      fill: '#99CCFF', width: 3, opacity: 1, dashArray: '2,3'
    }],
    //...
  }]
  //...
});
```



#### Types of Trendline

EjChart supports the following type of Trendlines.

- Linear
- Exponential
- Logarithmic
- Power
- Polynomial
- MovingAverage

#### Linear

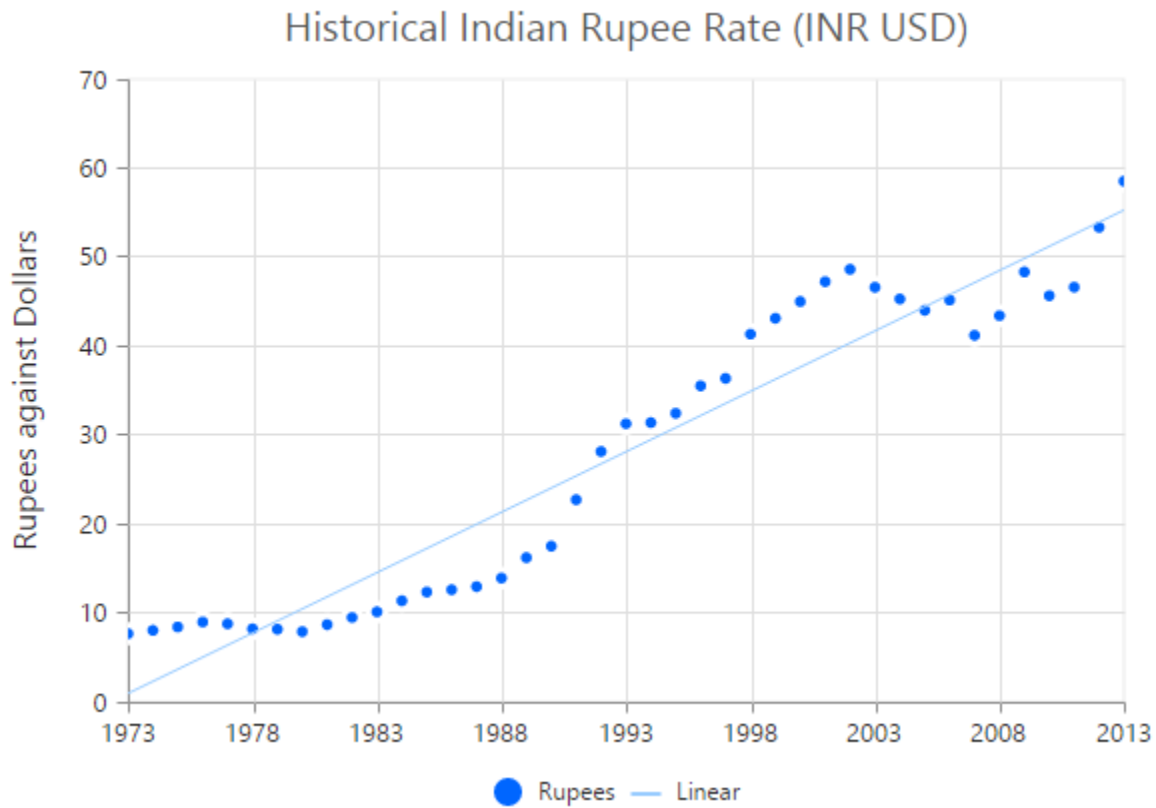
To render Linear Trendline, you have to set the [type](#) as “linear”.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  series:[{
    trendlines: [{
      //Change Trendline type
      type: "linear"
    }],
    //...
  }]
  //...
```



```
});
```

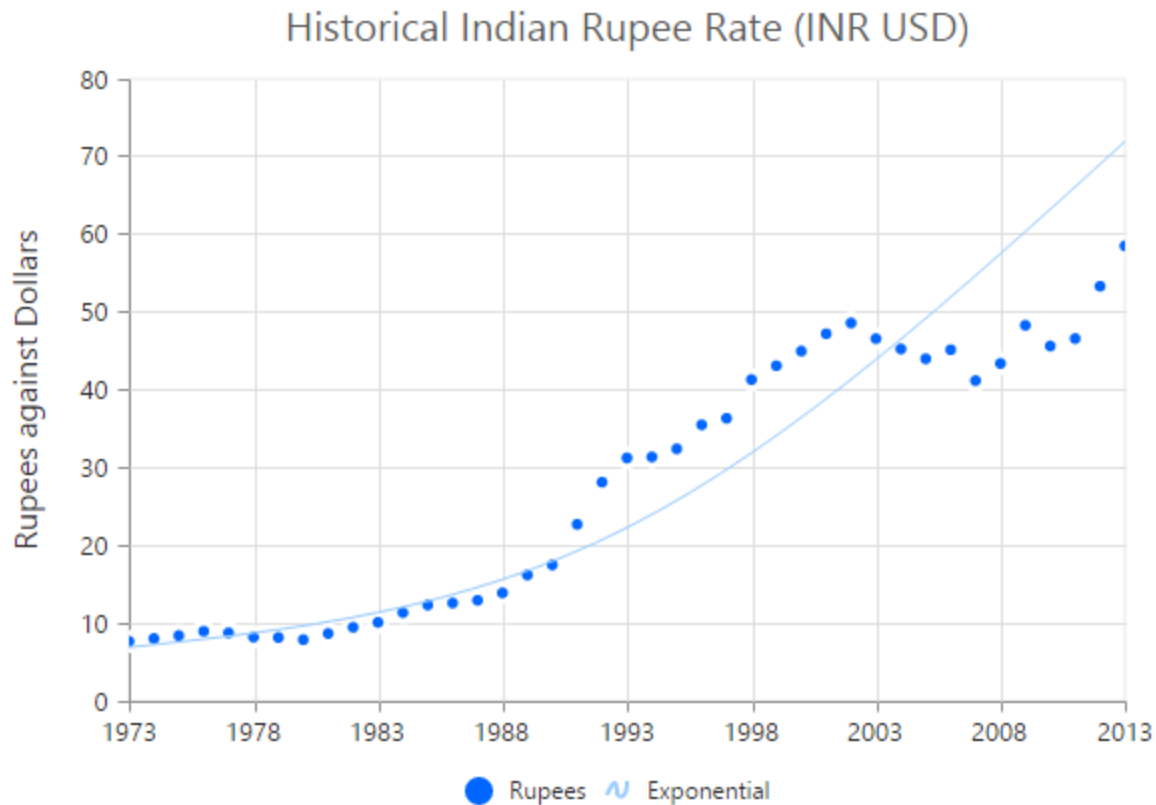


#### Exponential

Exponential Trendline can be rendered by setting the [type](#) as “**exponential**”.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  series:[{
    trendlines: [{
      //Change Trendline type
      type: "exponential"
    }],
    //...
  }]
  //...
});
```

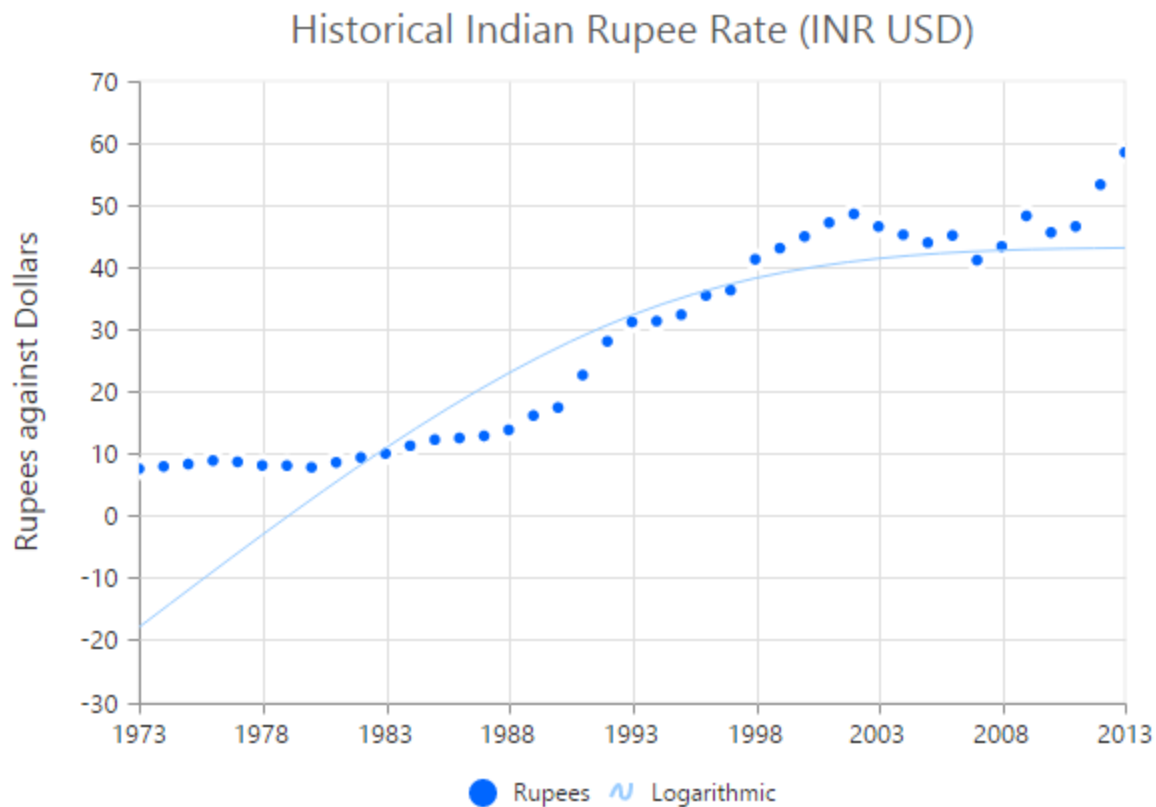


#### Logarithmic

Logarithmic Trendline can be rendered by setting the [type](#) as “**Logarithmic**”.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  series:[{
    trendlines: [{
      //Change Trendline type
      type: "logarithmic"
    }],
    //...
  }]
  //...
});
```

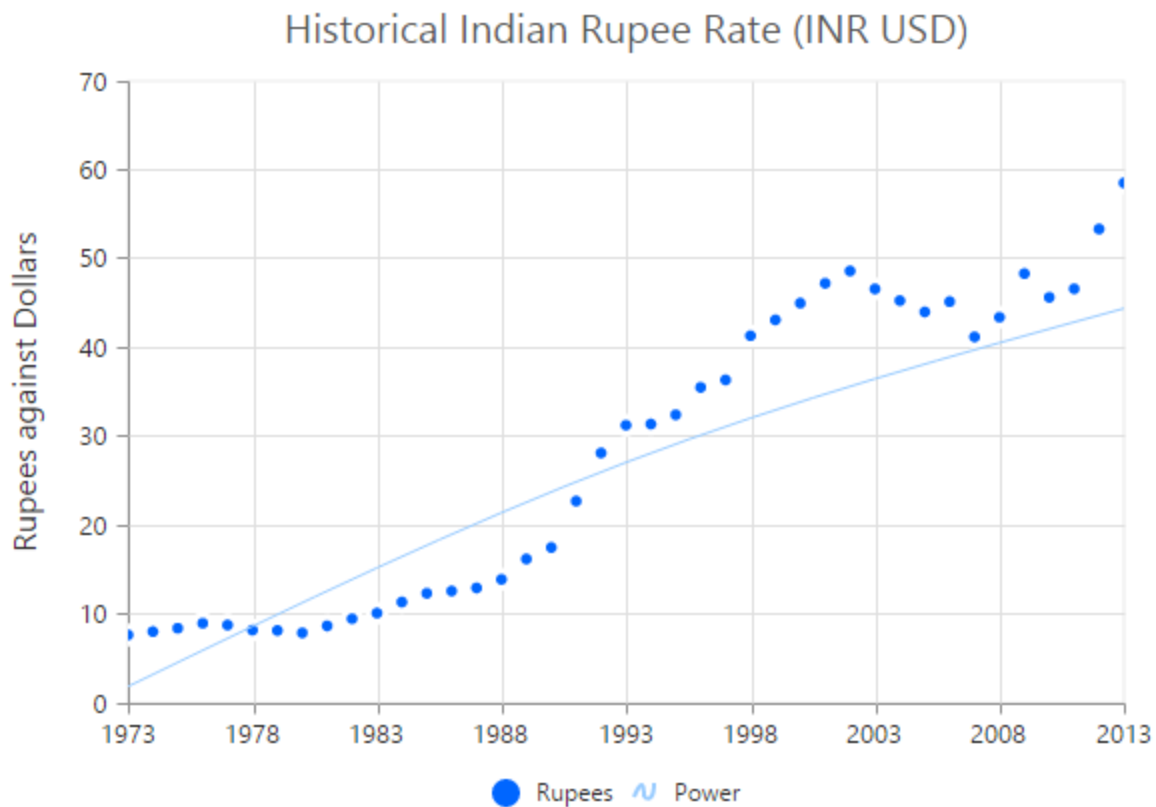


#### Power

Power Trendline can be rendered by setting the [type](#) of the trendline as “power”.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  series:[{
    trendlines: [{
      //Change Trendline type
      type: "power"
    }],
    //...
  }]
  //...
});
```

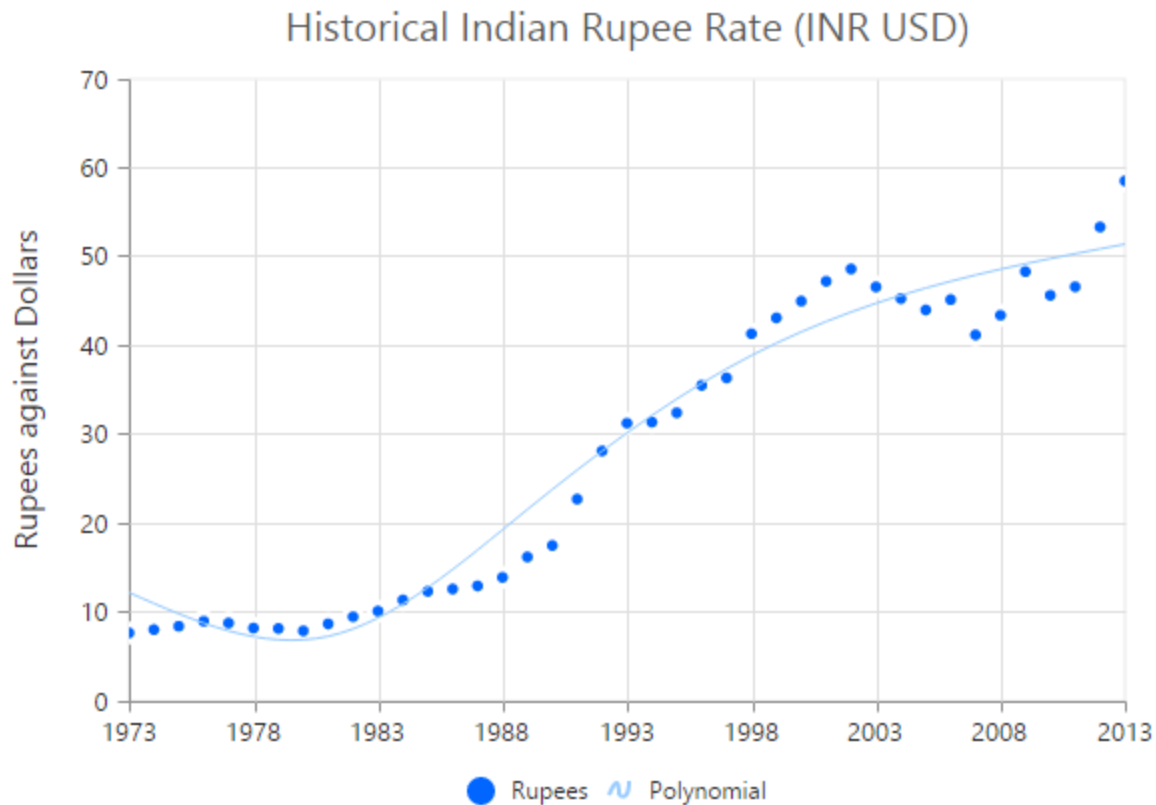


#### Polynomial

Polynomial Trendline can be rendered by setting the trendline [type](#) as “**polynomial**”. You can change the polynomial order by using the **polynomialOrder** of the trendlines. It ranges from 2 to 6.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  series:[{
    trendlines: [{
      //Change Trendline type
      type: "polynomial"
    }],
    //...
  }]
  //...
});
```

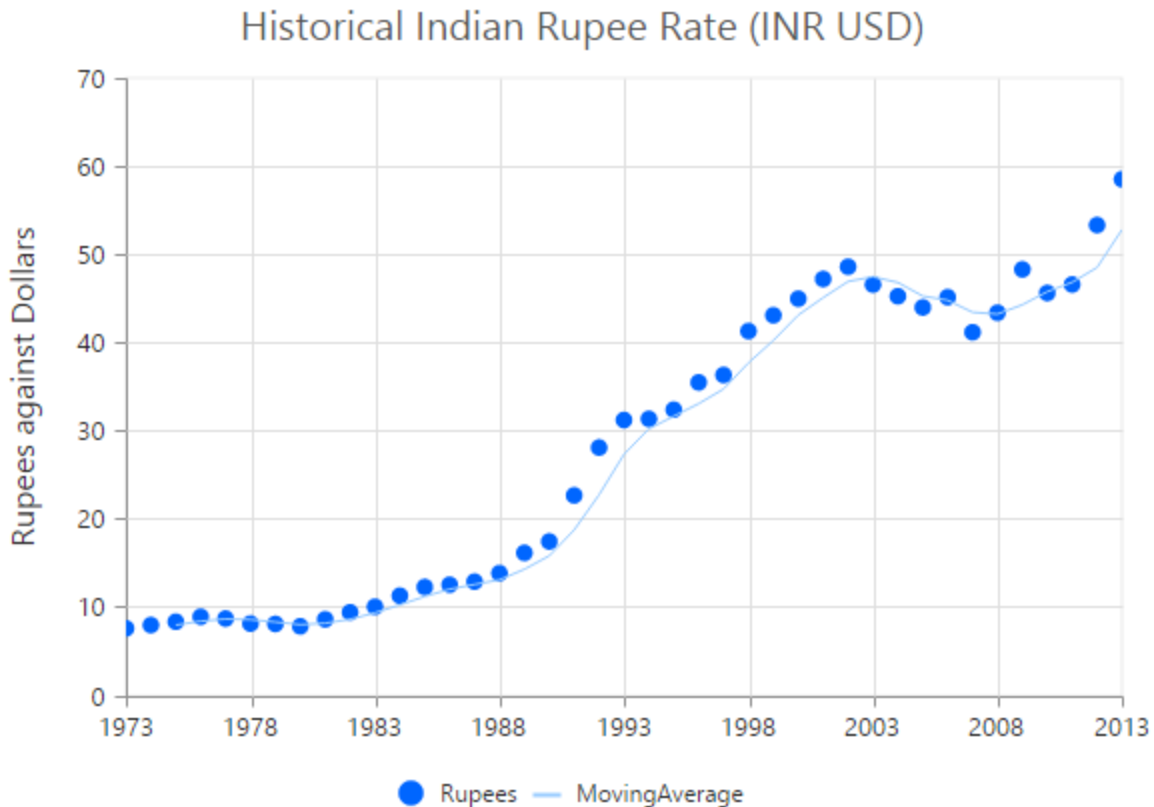


#### MovingAverage

MovingAverage Trendline can be rendered by setting the [type](#) of the trendline as “movingAverage”.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  series:[{
    trendlines: [{
      //Change Trendline type and set [`period`] (../api/ejchart#members:series-
      trendlines-period) for moving average
      type: "movingAverage", period: 3
    }],
    //...
  }]
  //...
});
```



#### Forecasting

**Trendline forecasting** is the prediction of future/past situations. **Forecasting** can be classified into two types as follows.

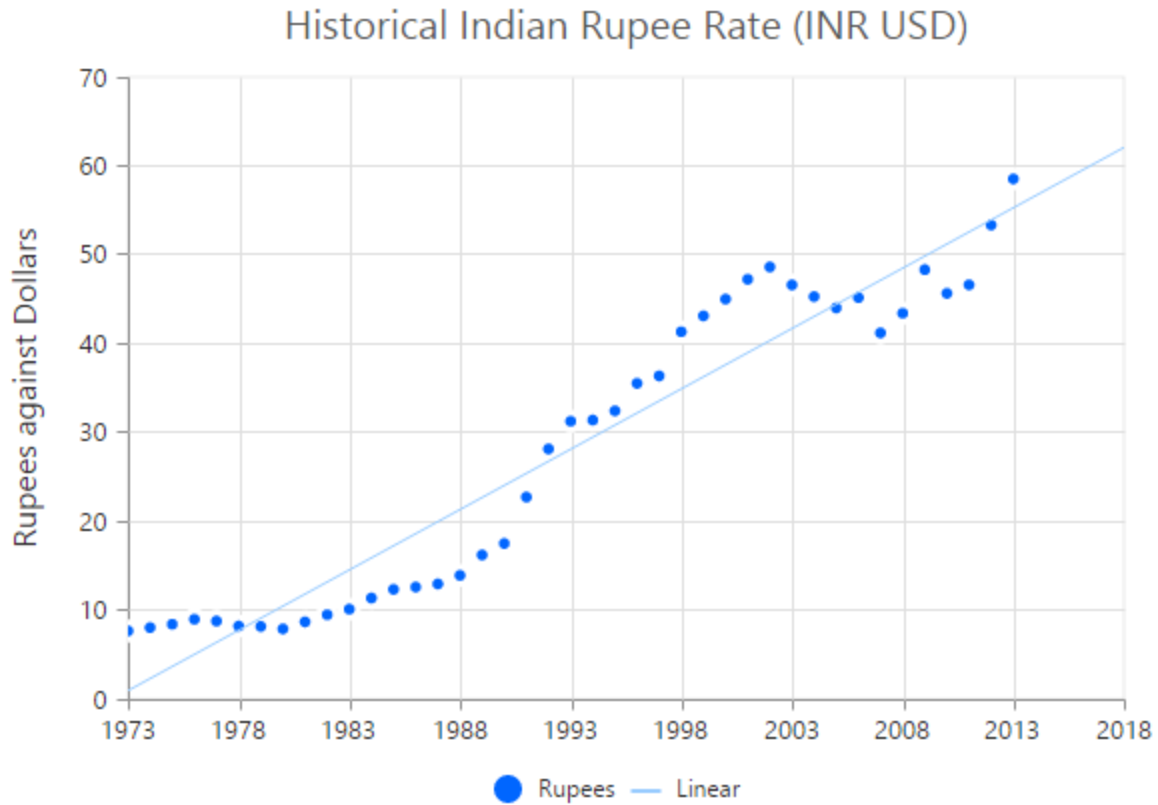
- Forward Forecasting
- Backward Forecasting

#### Forward Forecasting

The value set for [forwardForecast](#) is used to determine the distance moving towards the future trend.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  series:[{
    trendlines: [{
      //...
      //Set forward forecasting value
      forwardForecast: 5
    }],
    //...
  }]
  //...
});
```

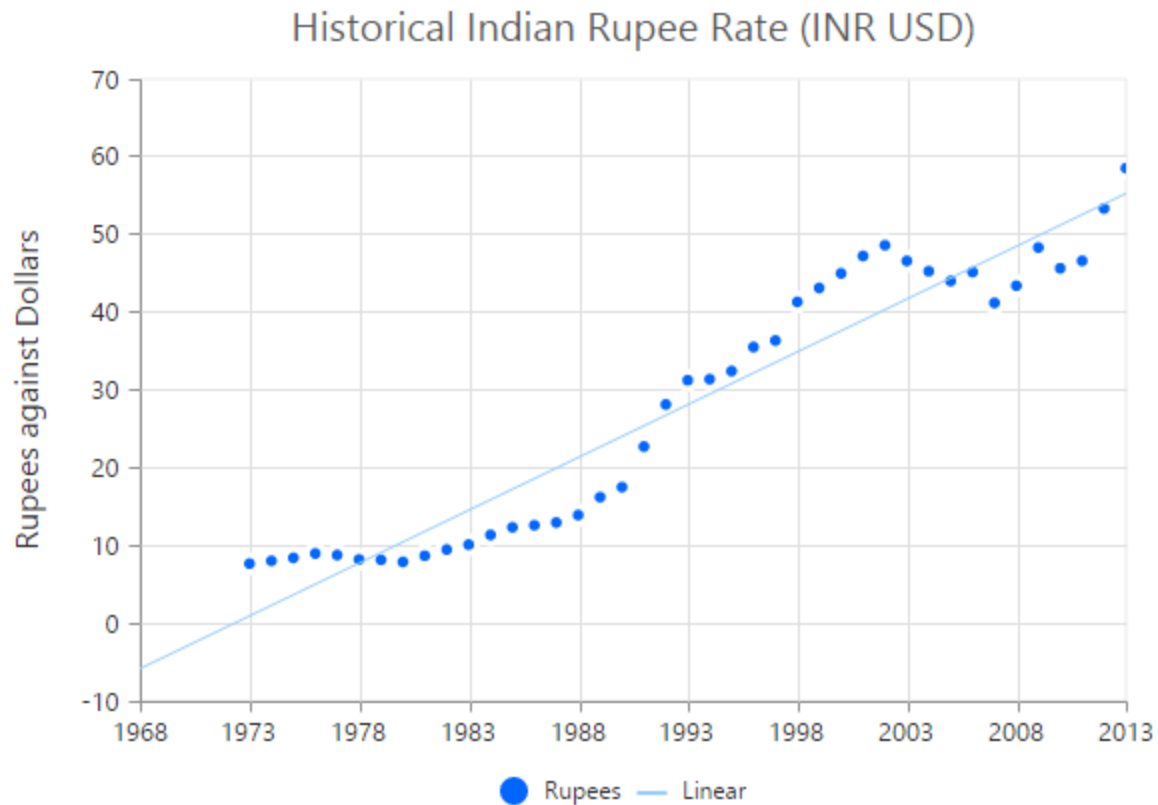


#### Backward Forecasting

The value set for the [backwardForecast](#) is used to determine the past trends.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  series:[{
    trendlines: [{
      //...
      //Set forward forecasting value
      backwardForecast: 5
    }],
    //...
  }]
  //...
});
```



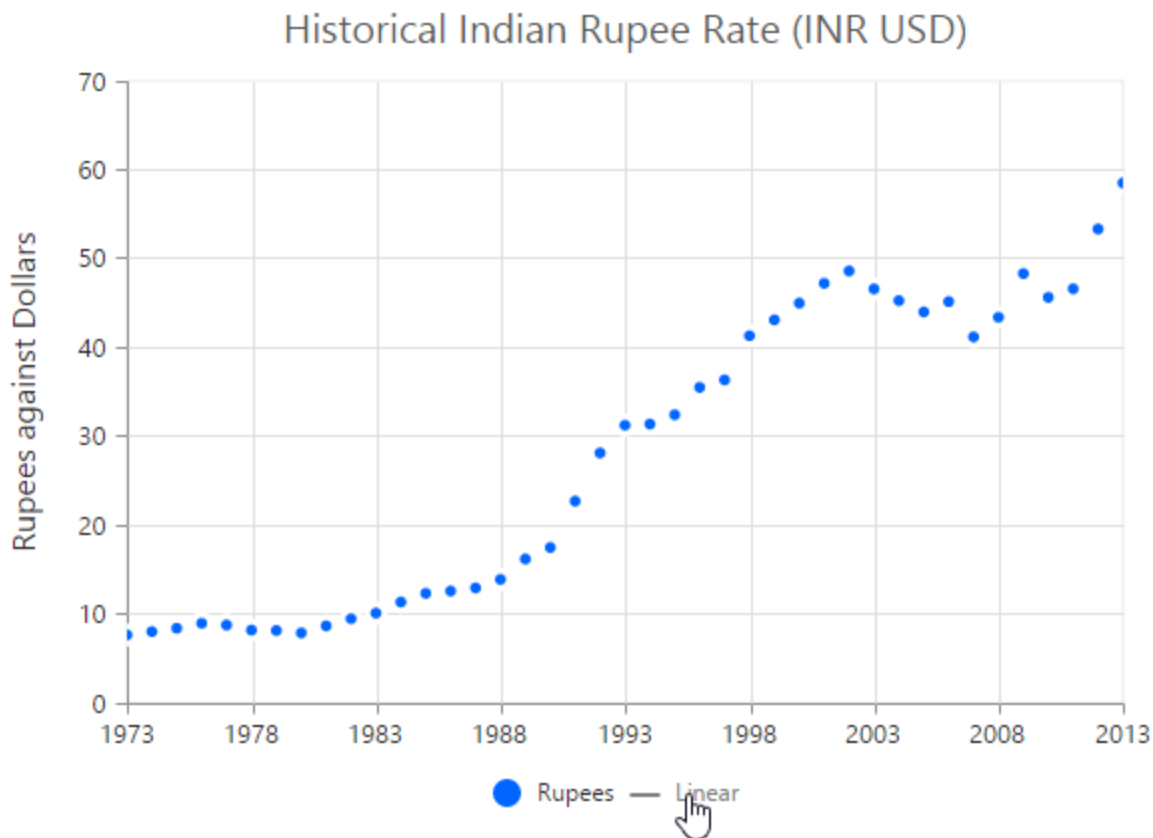
#### Trendlines Legend

To display the legend item for trendline, use the [name](#) property. You can interact with the trendline legends similar to the series legends (*show/hide trendlines on legend click*).

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  series:[{
    trendlines: [{
      //...
      //Set Trendline name to display in the legend
      name: 'Linear'
    }],
    //...
  }]
  //...
});
```





## Technical Indicators

EjChart control supports 10 types of technical indicators.

Bind data to render the indicator

You can bind the series [dataSource](#) to the indicator by setting the specific series name to the indicator by using the [indicators.seriesName](#) property.

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
$(function () {
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
//Initializing Series
series:[{
dataSource: chartData,
xName: "xDate",
high: "High",
low: "Low",
open: "Open",
close: "Close",
//Set name to series
name: 'Hilo',
// ...

```

```

}],
//Initializing Indicators
indicators: [{
//Set HiLo series dataSource to indicator using seriesName
seriesName: "Hilo",
// ...
}],
// ...
});
});
}

```

Also, you can add data to the indicator directly by using the [dataSource](#) option of the indicator.

### JAVASCRIPT

```

var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
//Initializing Indicators
indicators: [{
//Add dataSource to indicator directly
dataSource: chartData,
xName: "xDate",
high: "High",
low: "Low",
open: "Open",
close: "Close",
// ...
}],
// ...
});

```

### Indicator Types

#### Accumulation Distribution

To create an Accumulation Distribution indicator, set the [indicators.type](#) as “**accumulationDistribution**”. Accumulation Distribution require ‘**volume**’ field additionally with the [dataSource](#) to calculate the signal line.

### JAVASCRIPT

```

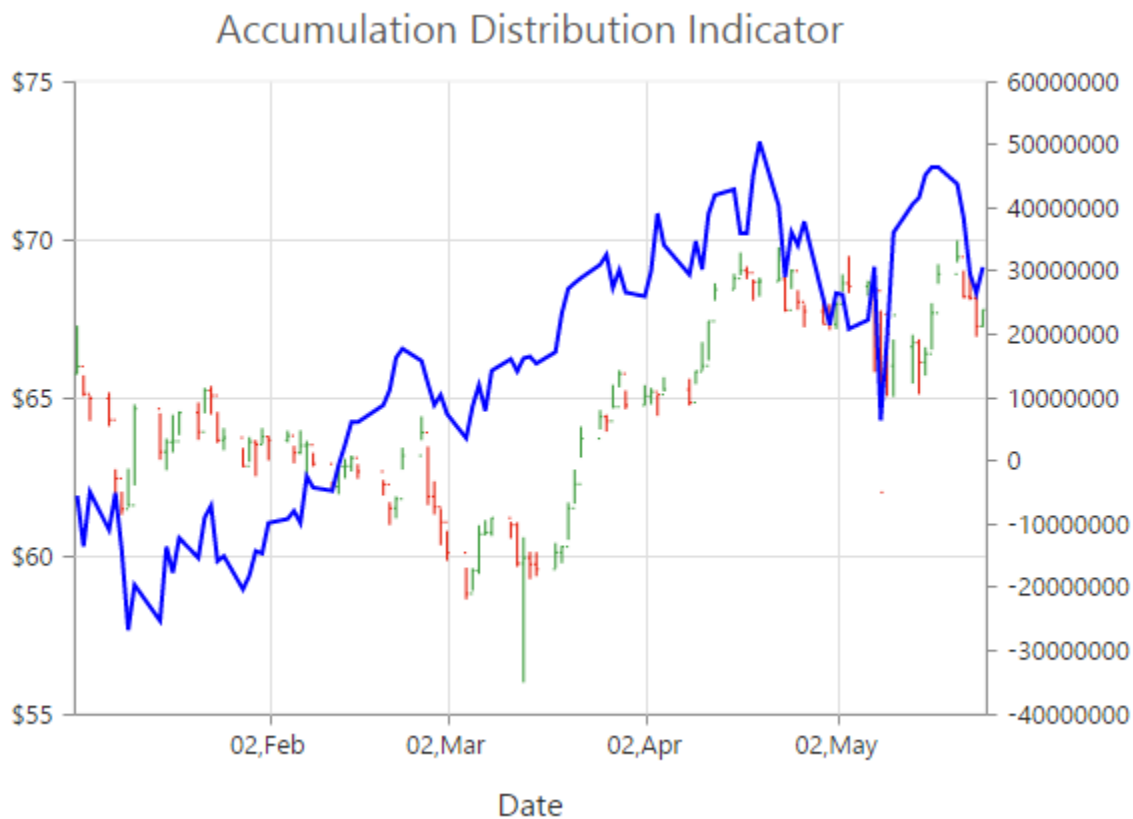
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// Initializing Series
series:[{
name: "Hilo",
dataSource: chartData,
xName: "xDate",
high: "High",
low: "Low",
open: "Open",
close: "Close",
//Add additional volume field to data source for accumulation distribution
volume: "Volume",
// ...
}],
//Initializing Indicators

```

```

indicators: [{
  seriesName: "Hilo",
  //Set indicator type
  type: "accumulationDistribution",
  // ...
}],
// ...
});

```



#### Average True Range (ATR)

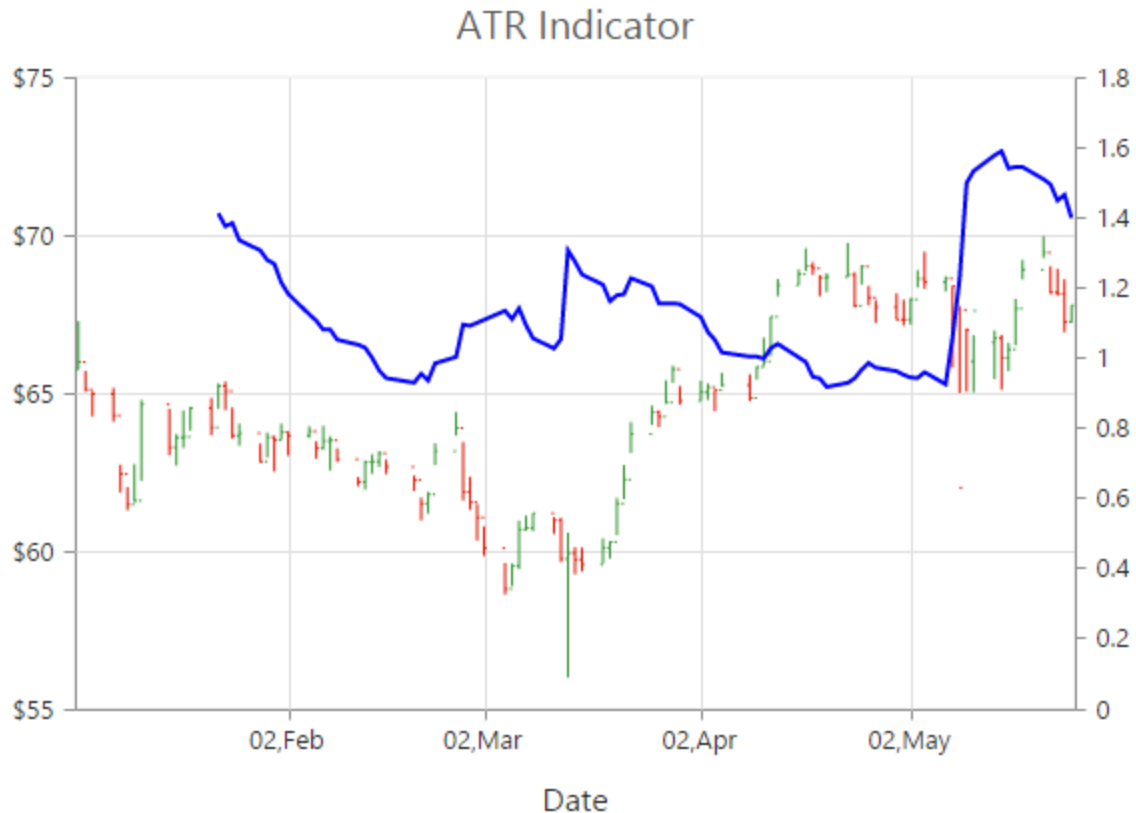
You can create an ATR indicator by setting the [indicators.type](#) as “ATR” in the [indicators](#).

#### JAVASCRIPT

```

var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  //Initializing Indicators
  indicators: [{
    //Set indicator type
    type: "ATR",
    // ...
  }],
  // ...
});

```

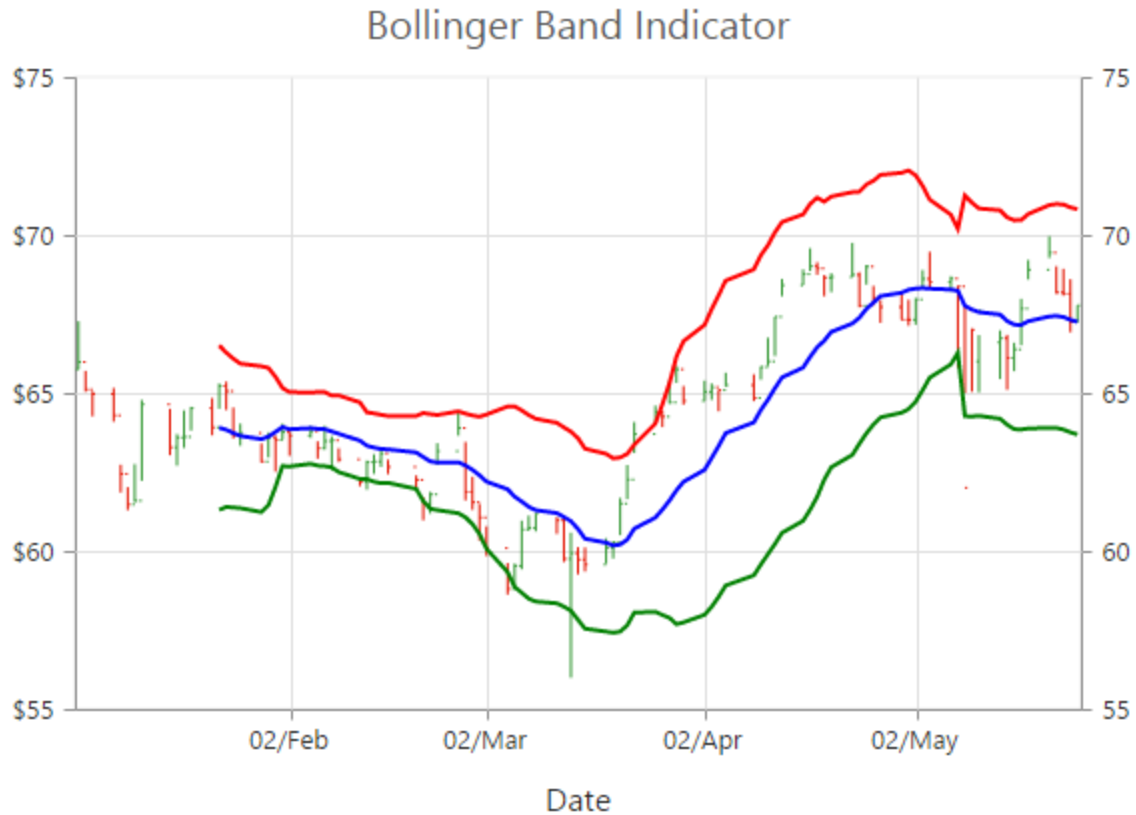


#### Bollinger Band

Bollinger Band indicator is created by setting the [indicators.type](#) as “**bollingerBand**”. It contains three lines, namely upper band, lower band and signal line. Bollinger Band default value of the period is 14 and standardDeviations is 2.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  //Initializing Indicators
  indicators: [{
    //Set indicator type
    type: " bollingerBand",
    // ...
  }],
  // ...
});
```

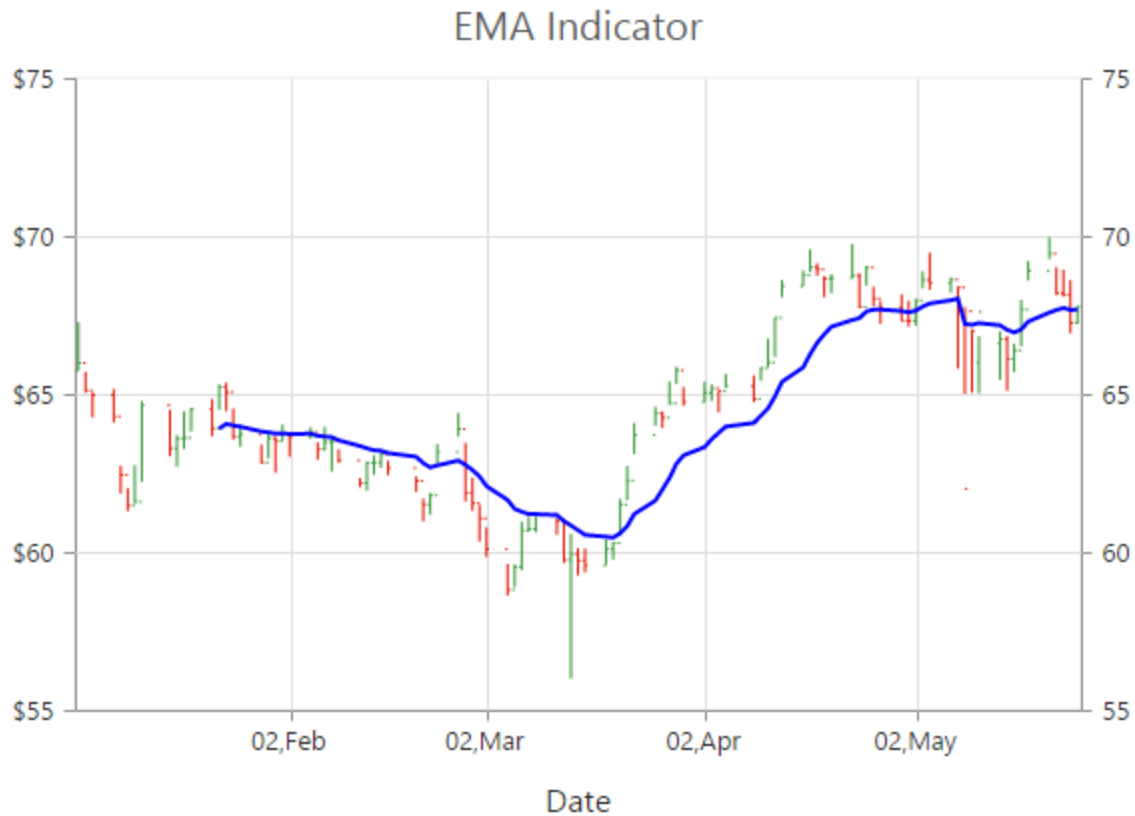


#### Exponential Moving Average (EMA)

To render an EMA indicator, you have to set the [indicators.type](#) as "EMA".

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  //Initializing Indicators  
  indicators: [{  
    //Set indicator type  
    type: "EMA",  
    // ...  
  }],  
  // ...  
});
```

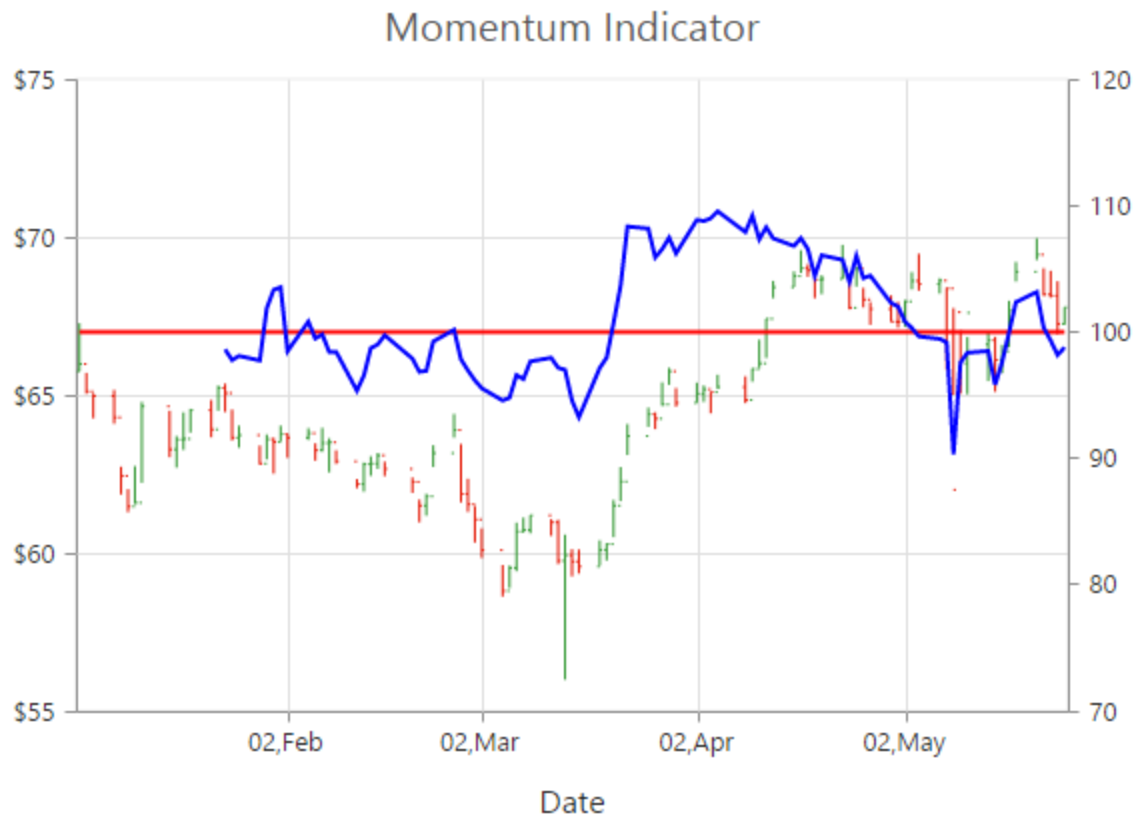


#### Momentum

Momentum Technical indicator is created by setting the [indicators.type](#) as “**momentum**”. The momentum indicator renders two lines, namely upper band and signal line. Upper band always rendered at the value 100 and the signal line is calculated based on the momentum of the data.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  //Initializing Indicators  
  indicators: [{  
    //Set indicator type  
    type: "momentum",  
    // ...  
  }],  
  // ...  
});
```

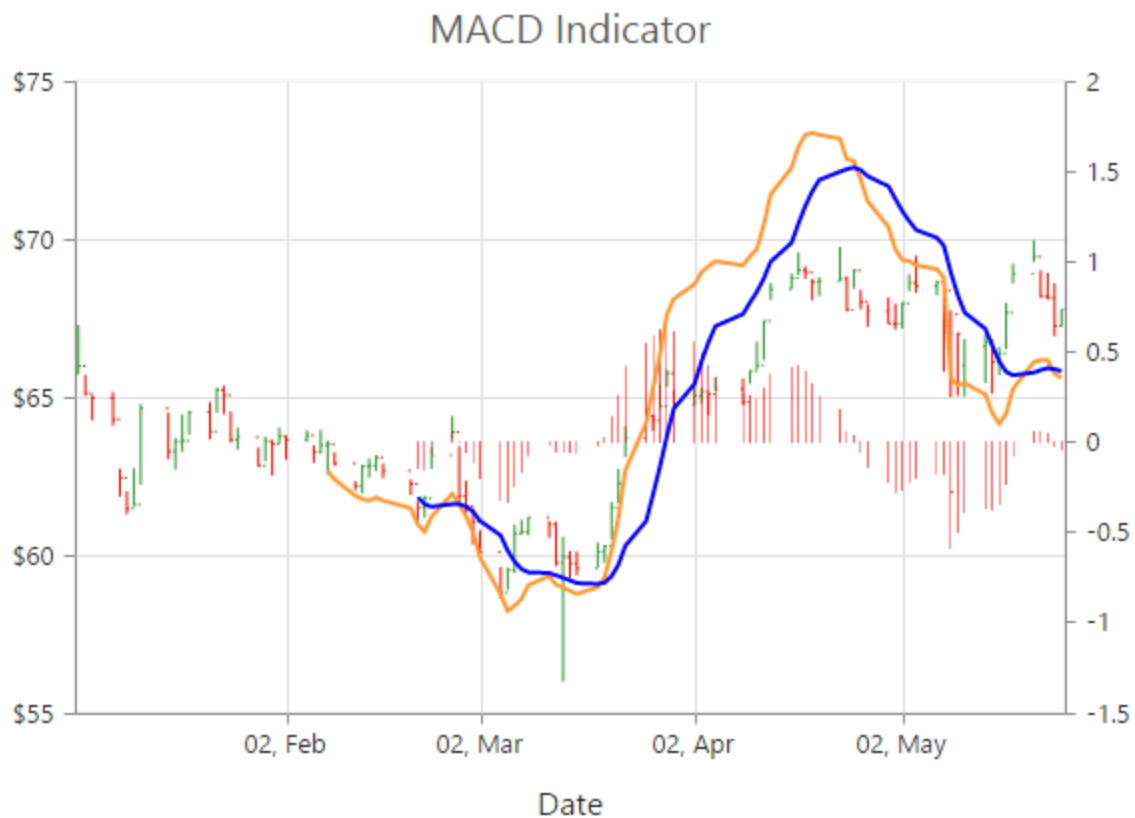


#### *Moving Average Convergence Divergence (MACD)*

To render an MACD indicator, you have to set the [indicators.type](#) as “**macd**”. MACD indicator contains MACD line, Signal line and Histogram column. Histogram is used to differentiate MACD and signal line.

#### **JAVASCRIPT**

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  //Initializing Indicators
  indicators: [{
    //Set indicator type
    type: "macd",
    // ...
  }],
  // ...
});
```



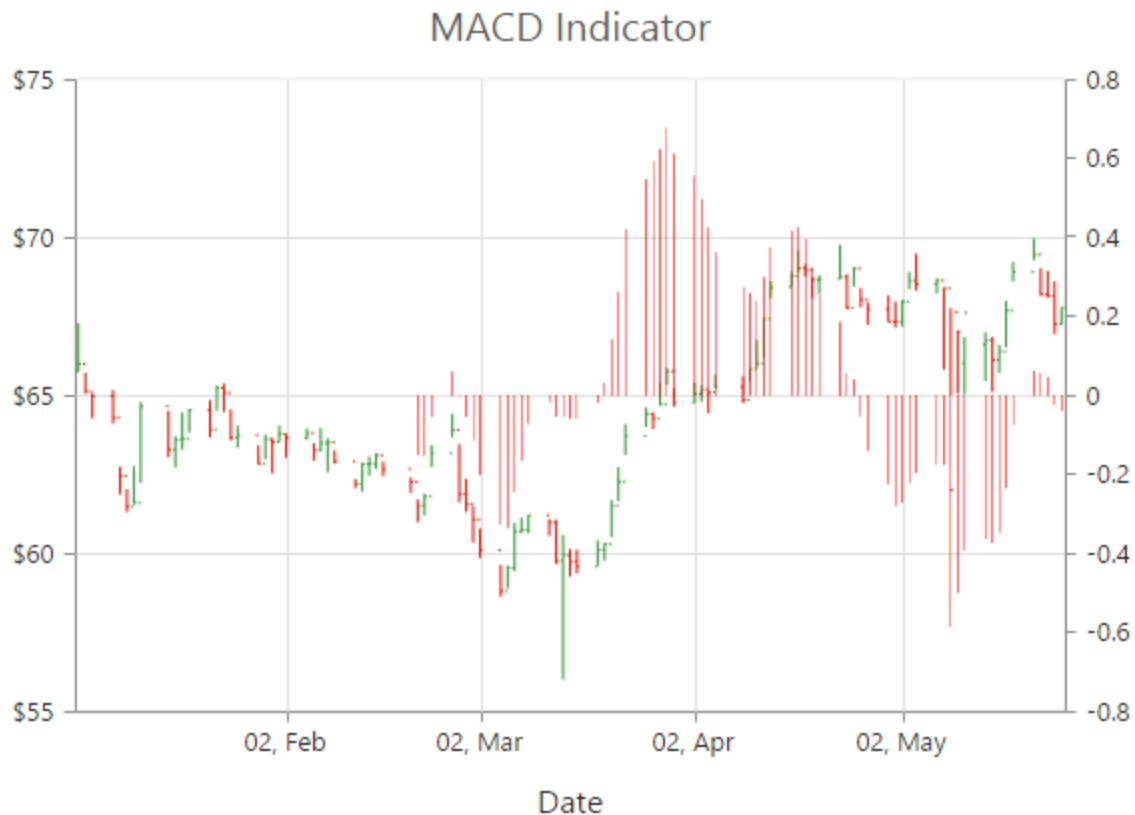
#### macdType

By using the [macdType](#) enumeration property, you can change the MACD rendering as *line*, *histogram* or *both*.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  //Initializing Indicators
  indicators: [{
    type: "macd",
    //Set macd draw type
    macdType: "histogram",
    // ...
  }],
  // ...
});
```



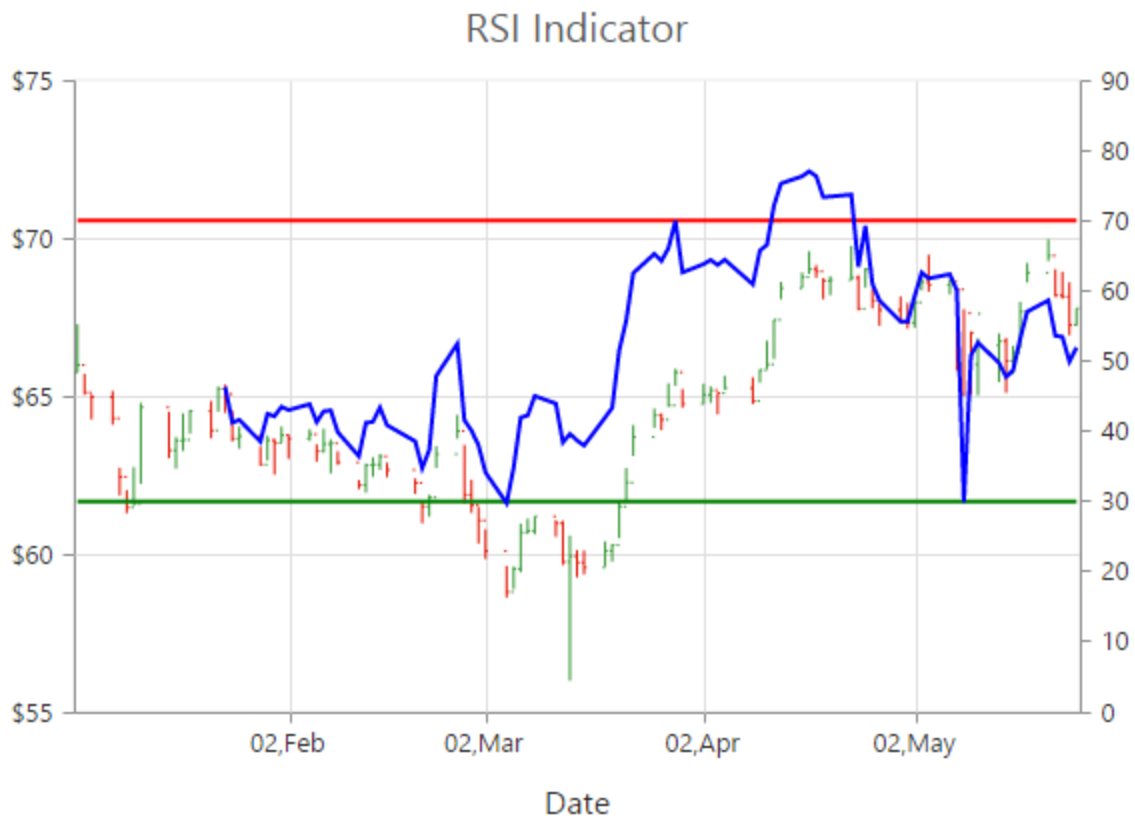


#### Relative Strength Index (RSI)

To render the RSI indicator, set the [indicators.type](#) as **"RSI"**. It contains three lines, namely upper band, lower band and signal line. Upper and lower band always render at value 70 and 30 respectively and signal line is calculated based on the **RSI** formula.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  //Initializing Indicators
  indicators: [{
    //Set indicator type
    type: "RSI",
    // ...
  }],
  // ...
});
```

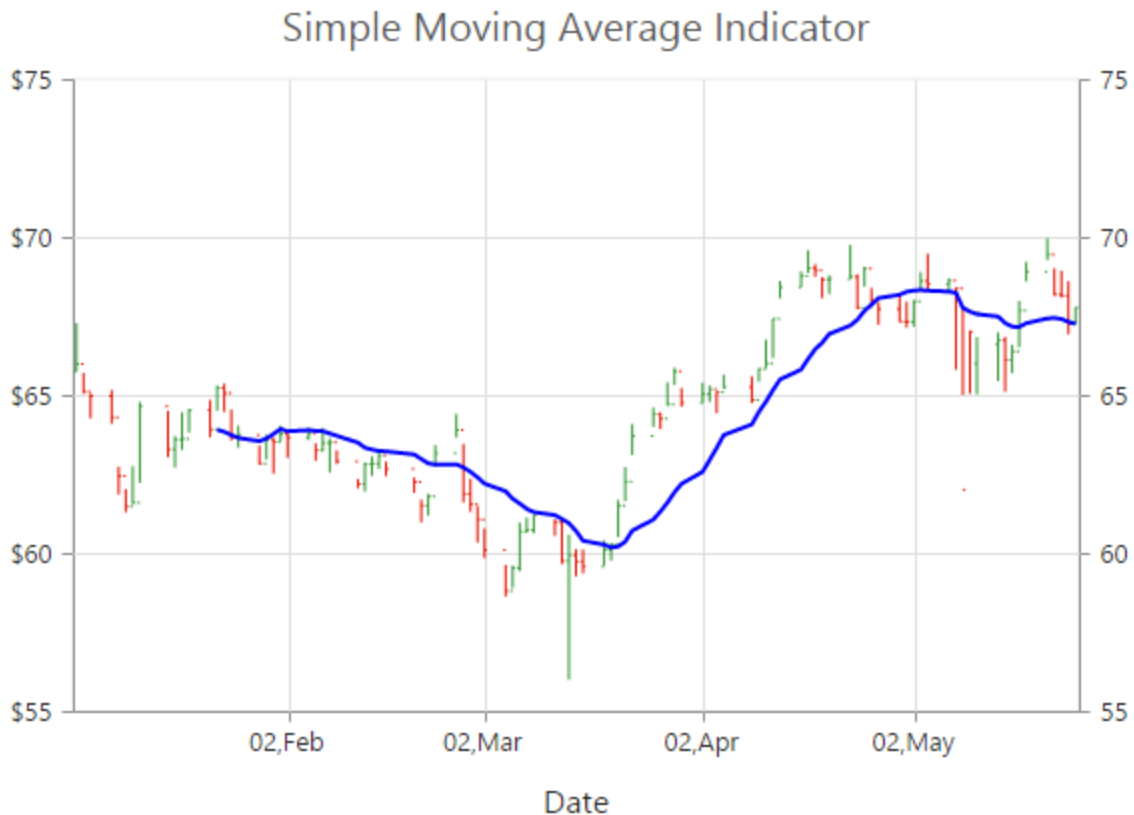


#### Simple Moving Average (SMA)

To render the SMA indicator, you should specify the [indicators.type](#) as "SMA".

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  //Initializing Indicators
  indicators: [{
    //Set indicator type
    type: "SMA",
    // ...
  }],
  // ...
});
```

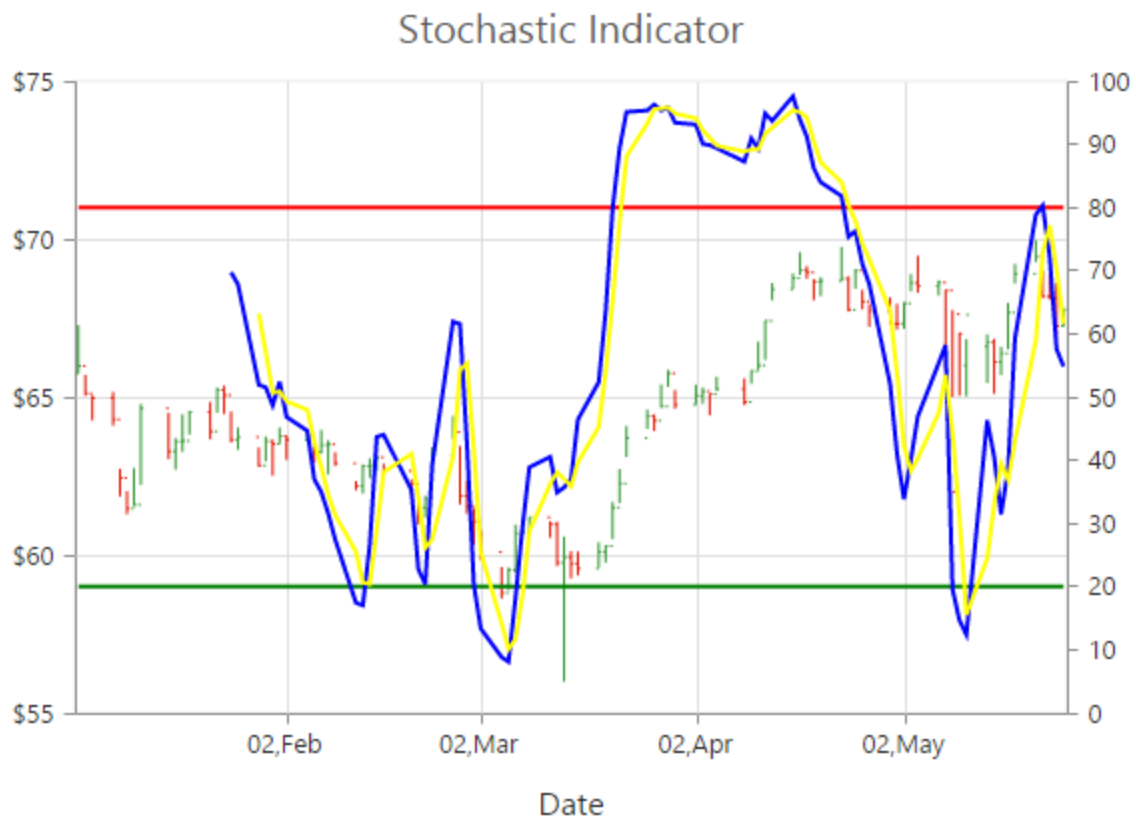


#### Stochastic

For the Stochastic indicator, you need to set the [indicators.type](#) as **"stochastic"**. The Stochastic indicator renders four lines namely, upper line, lower line, stochastic line and the signal line. Upper line always rendered at value 80 and the lower line is rendered at value 20. Stochastic and Signal Lines are calculated based on the stochastic formula.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  //Initializing Indicators  
  indicators: [{  
    //Set indicator type  
    type: "stochastic",  
    // ...  
  }],  
  // ...  
});
```

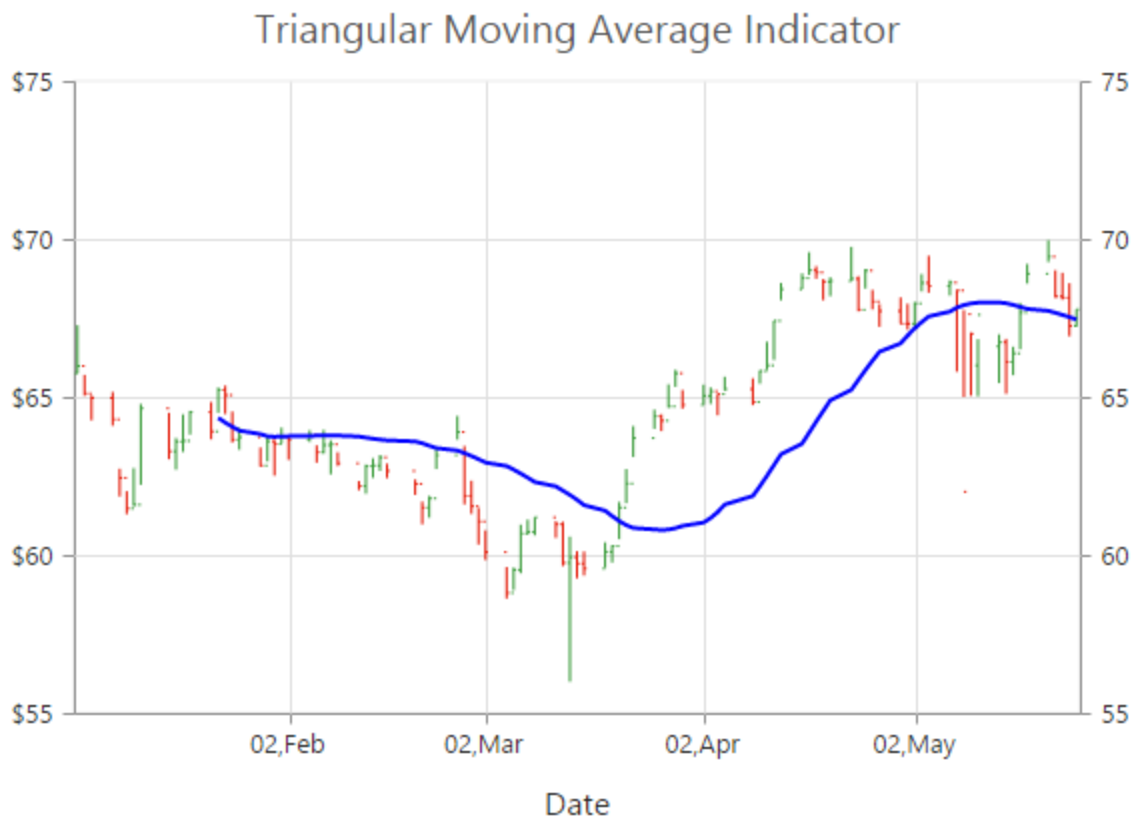


#### Triangular Moving Average (TMA)

To render the TMA indicator, you should specify the [indicators.type](#) as "TMA".

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  //Initializing Indicators
  indicators: [{
    //Set indicator type
    type: "TMA",
    // ...
  }],
  // ...
});
```

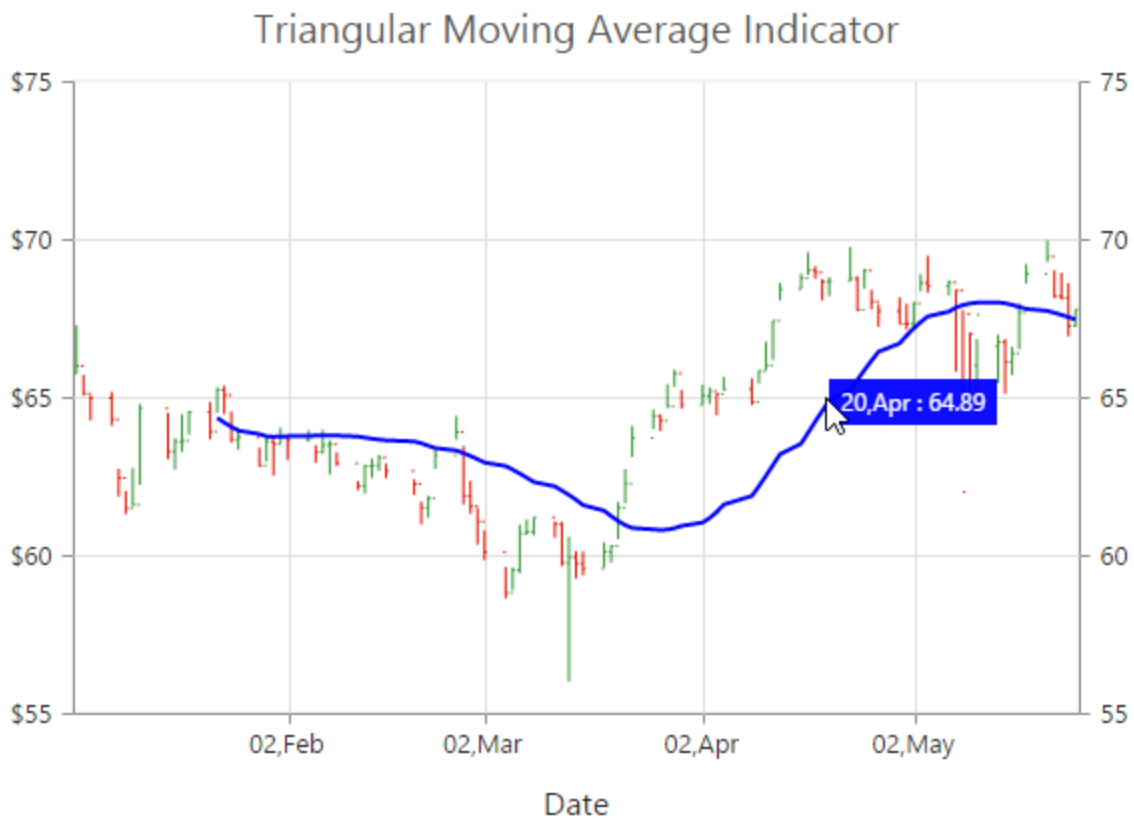


#### Enable Tooltip

To display the indicator tooltip, use [visible](#) option of the [indicators.tooltip](#). Also, you can change and customize the tooltip color, border, format and font properties similar to the series tooltip.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  //Initializing Indicators
  indicators: [{
    // ...
    tooltip: {
      //Enable tooltip for indicator
      visible: true
    },
  },
  ],
  // ...
});
```



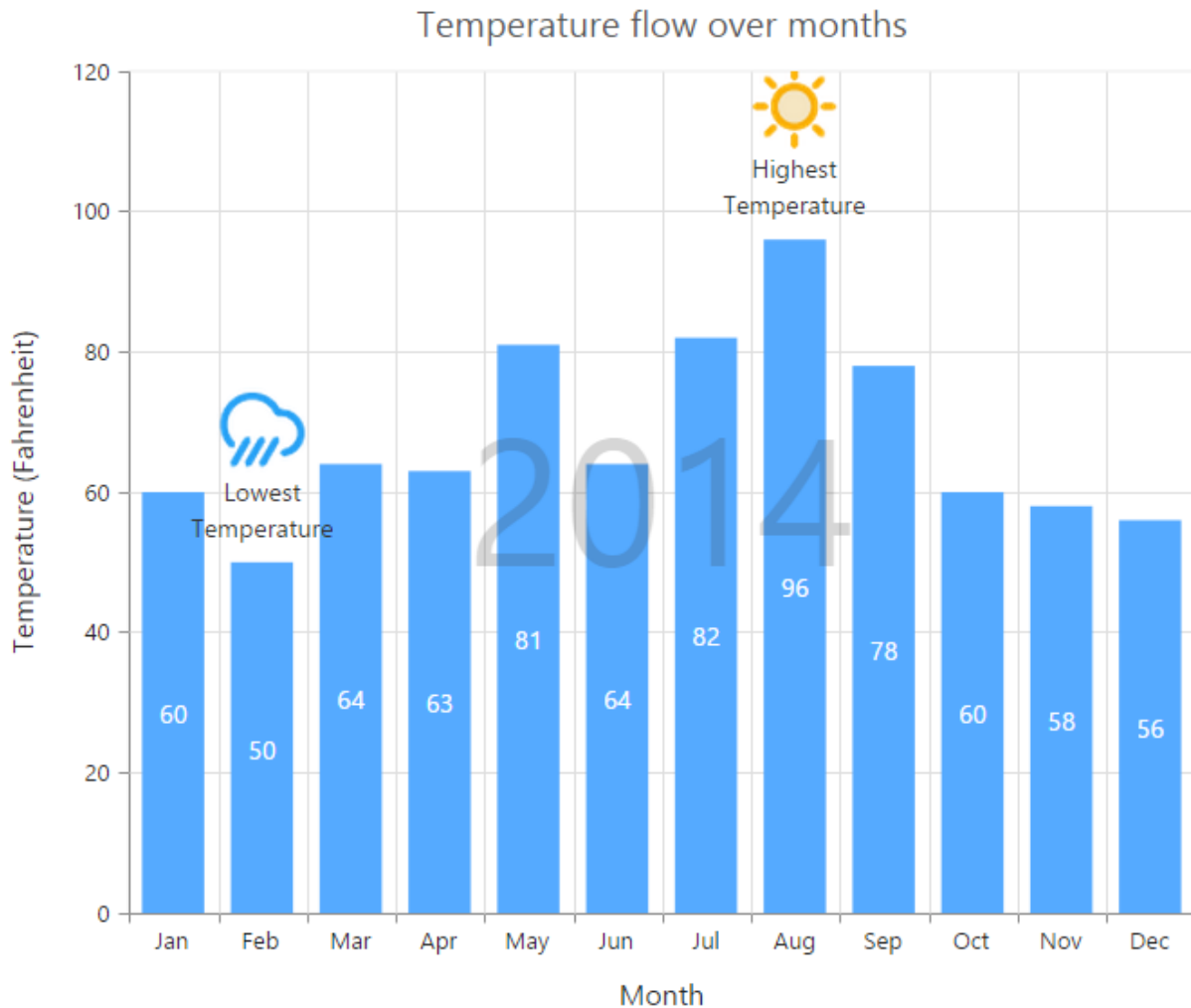
## Annotations

annotations are used to mark the specific area of interest in the chart area with texts, shapes or images.

You can add annotations to the chart by using the [annotations](#) option. By using the [content](#) option of annotation object, you can specify the id of the element that needs to be displayed in the chart area.

## HTML

```
<body>
<div id="chartcontainer"></div>
<div id= "watermark" style="font-size:100px; display:none">2014</div>
<script>
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
// ...
annotations: [
//Add Annotation content here
{ visible: true, content: "watermark", opacity: 0.2, region: "series" }
// ...
],
// ...
});
</script>
</body>
```



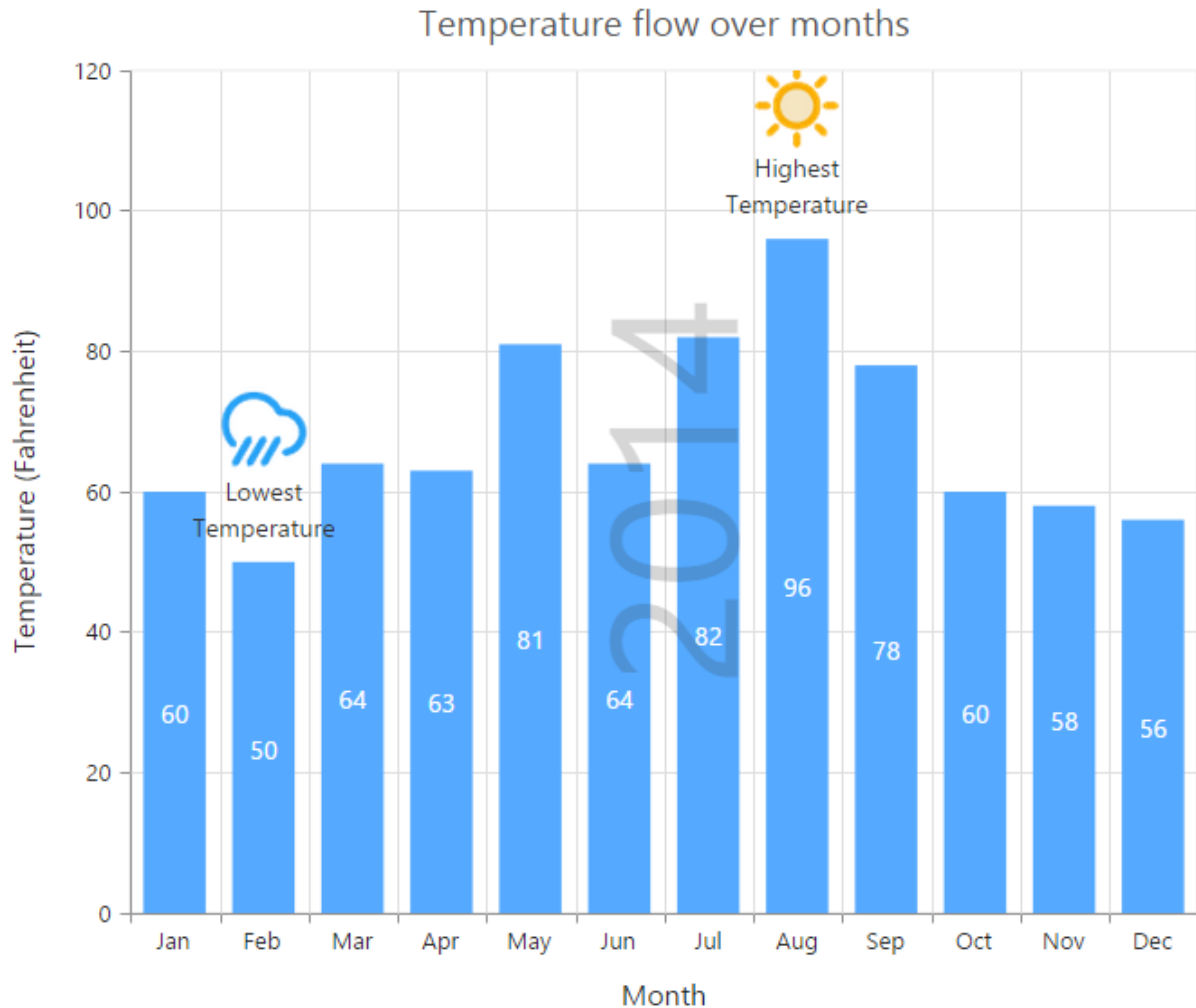
**Note:** Annotations are not supported in 3D chart types.

Rotate the annotation template

To rotate the annotation template, you can use the [angle](#) property of the annotations.

### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  annotations: [{ visible: true,
    content: "watermark",
    //Rotate the Annotation template
    angle: 270,
  },
  // ...
},
  // ...
});
```



### Positioning Annotation

You can position annotations either by using the coordinates ([x](#) and [y](#) options) or by using the alignment options ([horizontalAlignment](#) and [verticalAlignment](#)).

By using the [coordinateUnit](#) option, you can specify whether the value provided in the [x](#) and [y](#) options are relative to the chart or axis.

- If the [coordinateUnit](#) is set to none, the annotations are placed relative to the chart/plot area by using the [horizontalAlignment](#) and [verticalAlignment](#) options.
- If the [coordinateUnit](#) is set to points, the x and y values of the annotation are the coordinates relative to the axis and annotation is positioned relative to the axis. By default, the x and y values are associated with the [primaryXAxis](#) and [primaryYAxis](#). In case, when the chart contains multiple axis and you want to associate the annotation with a particular axis, you can specify the [xAxisName](#) and [yAxisName](#) options of the annotation object.
- If the [coordinateUnit](#) is set to pixels, the x and y values are coordinates relative to the top-left corner of the chart/plot area.

---

**Note:** By using the [region](#) option, you can specify whether the annotation is placed relative to the entire chart or plot area.

---

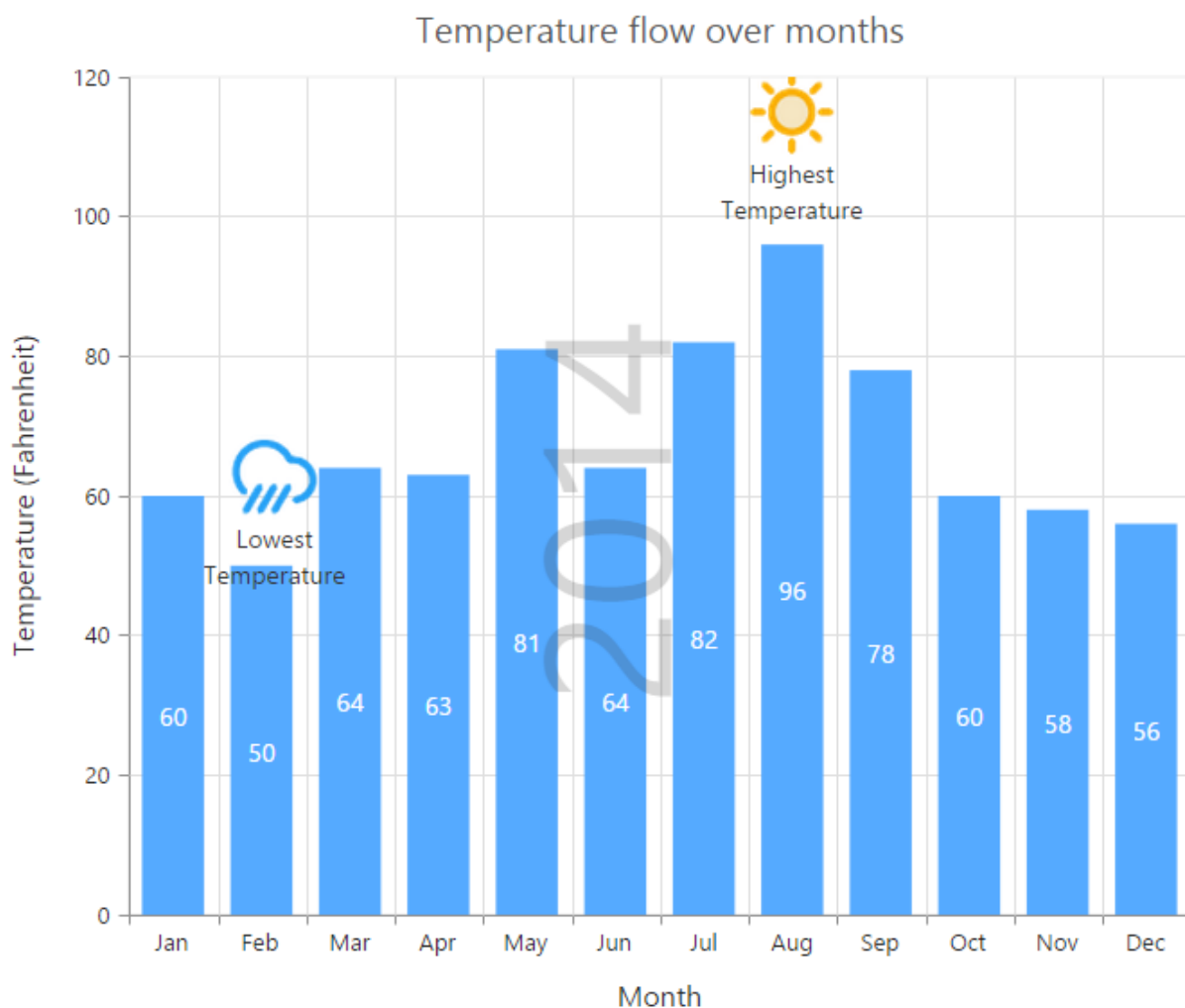


**JAVASCRIPT**

```

var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  annotations: [{ visible: true,
    content: "lowtemp",
    //Change coordinateUnit type to pixels
    coordinateUnit: "pixels", x: 170, y: 350,
    // ...
  },
  // ...
  ],
  // ...
});

```

**Annotation alignments**

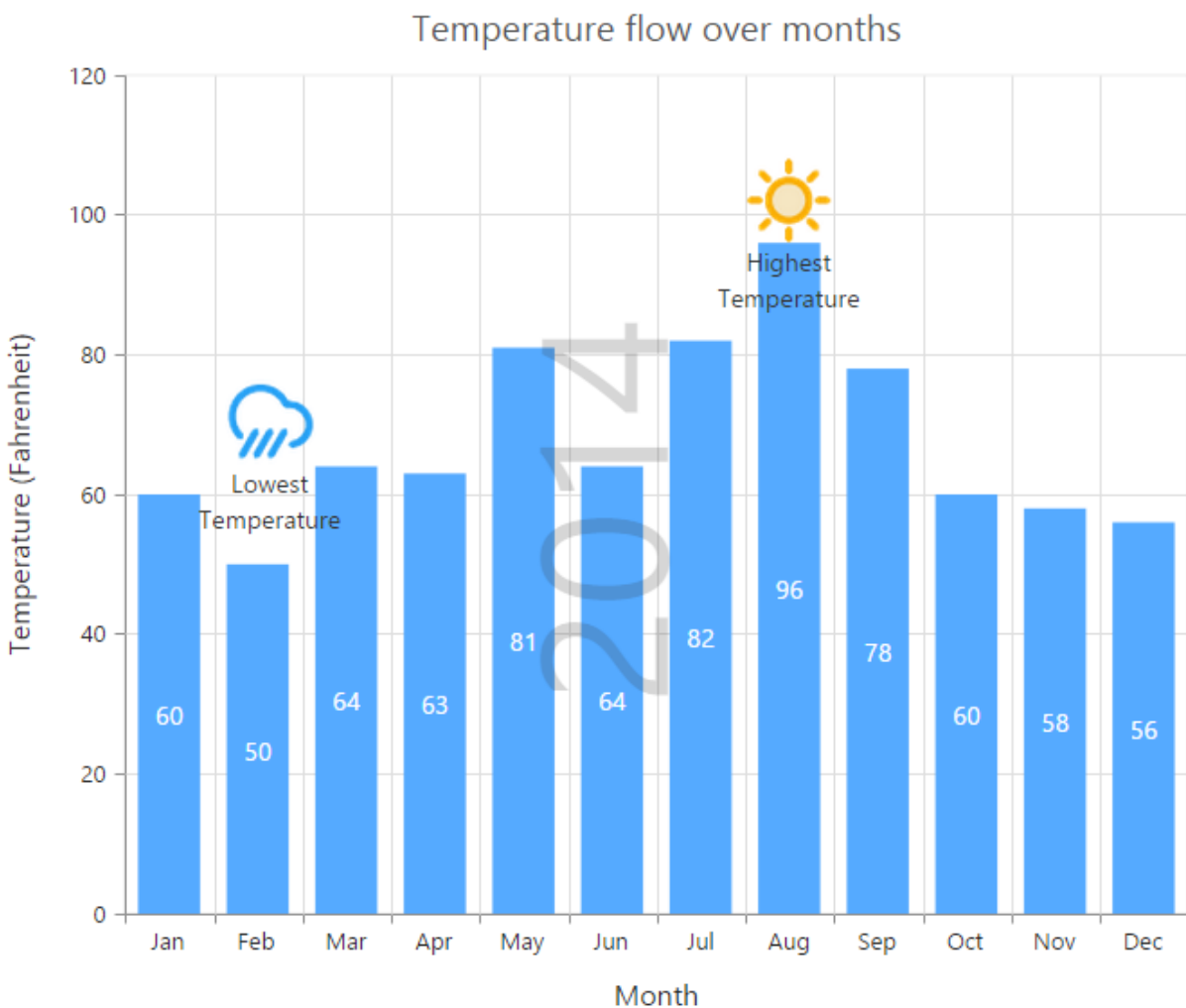
When the coordinateUnit is set to pixels or points, you can align the annotation relative to the coordinates by using the [horizontalAlignment](#) and [verticalAlignment](#) options.

**JAVASCRIPT**

```

var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  annotations: [{ visible: true,
    content: "hightemp",
    //Change alignment of annotation template
    verticalAlignment: "middle",
    horizontalAlignment: "near",
    margin: { right: 40 }
  },
  // ...
  ],
  // ...
});

```



## Appearance

### Custom Color Palette

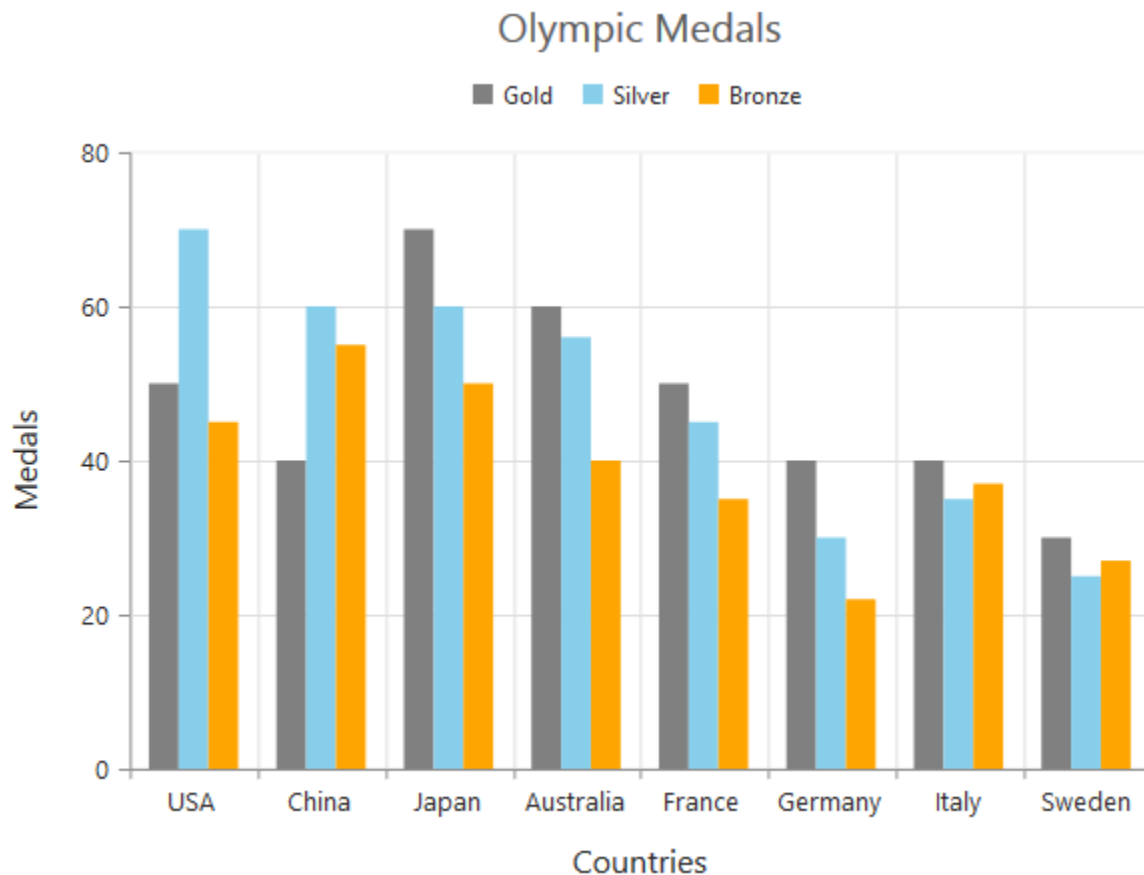
The Chart displays different series in different colors by default. You can customize the color of each series by providing a custom color palette of your choice by using the [palette](#) property.

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
  $(function () {
    var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
      //Providing a custom palette
      palette: [ "grey", "skyblue", "orange", ],
      // ...
    });
  });
}

```




---

**Note:** The Color palette is applied to the points in accumulation type series

---

#### Built-in Themes

Following are the built-in themes available in the Chart

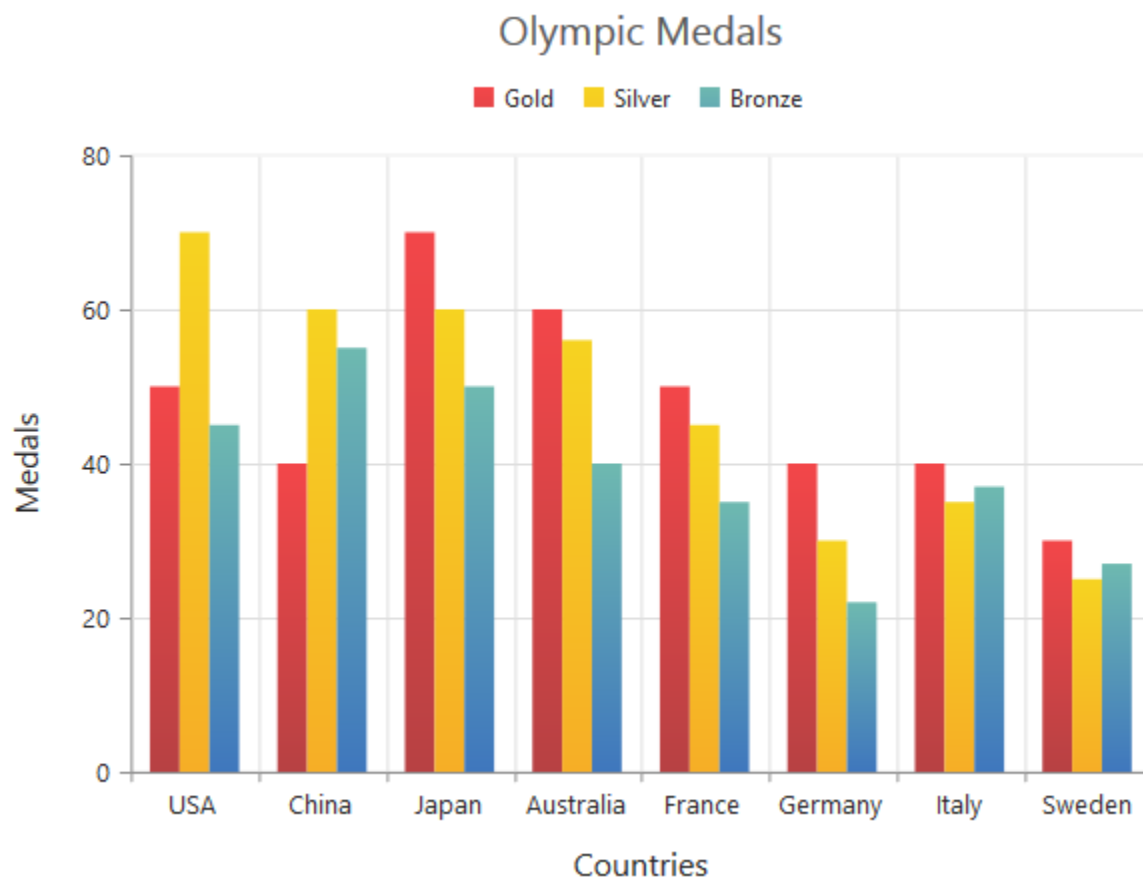
- flatLight
- flatDark
- gradientLight
- gradientDark
- azure
- azureDark

- lime
- limeDark
- saffron
- saffronDark
- gradient-azure
- gradient-azureDark
- gradient-lime
- gradient-limeDark
- gradient-saffron
- gradient-saffronDark

You can set your desired theme by using the [theme](#) property. Flat light is the default theme used in the Chart.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  //Using gradient theme  
  theme: "gradientLight",  
  // ...  
});
```

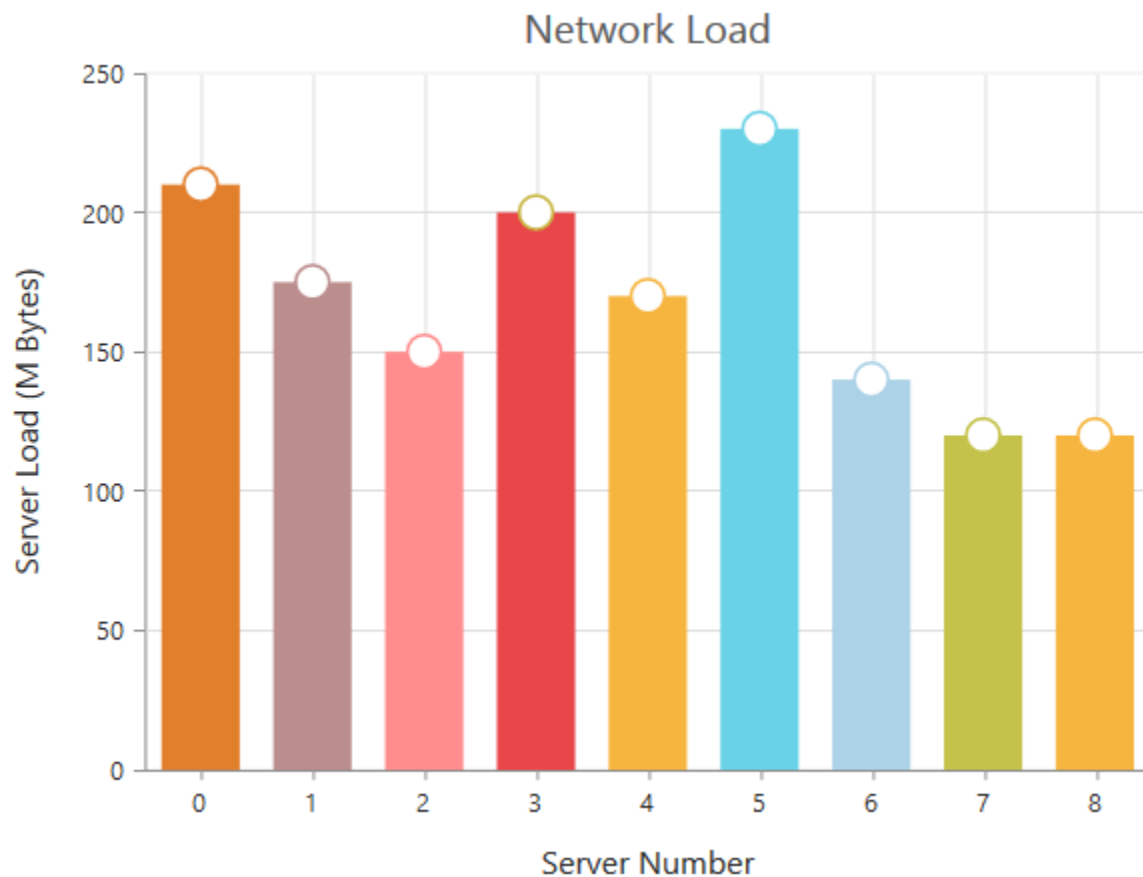


### Point level customization

Marker, data label and fill color of each point in a series can be customized individually by using the [points](#) collection.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  series: [{
    //Customizing marker and fill color of a point
    points: [
      {
        x : 0,
        y: 210,
        fill: "#E27F2D",
        marker: {
          visible: true,
          // ...
        }
      },
      // ...
    ],
    // ...
  }],
  // ...
});
```



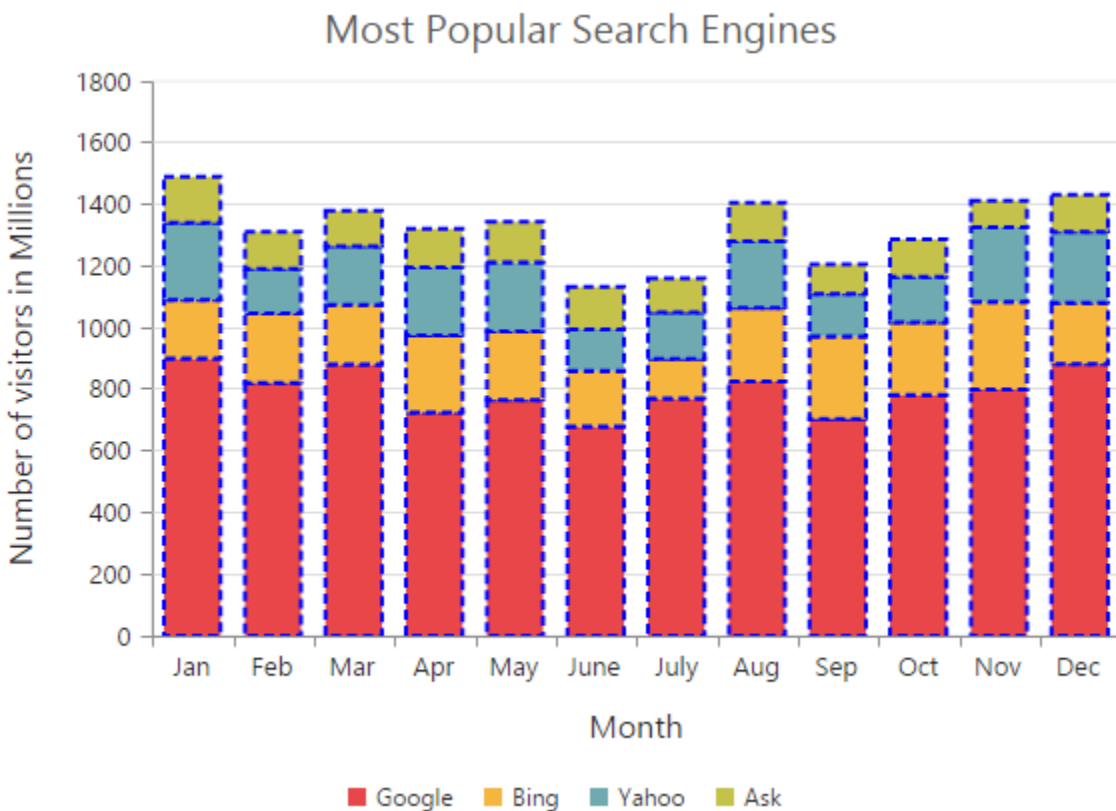
### Series border customization

To customize the series border color, width and dashArray, you can use [series.border](#) option.

**Note:** Series border can be applied to all the series (except Line, Spline, HiLo, HiLoOpenClose and StepLine series).

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  //...
  series: [{
    //Change the color, width and dashArray to customize the border of series
    border: { color: "blue", width: 2, dashArray: "5,3" }
    //...
  }]
  //...
});
```



### Chart area customization

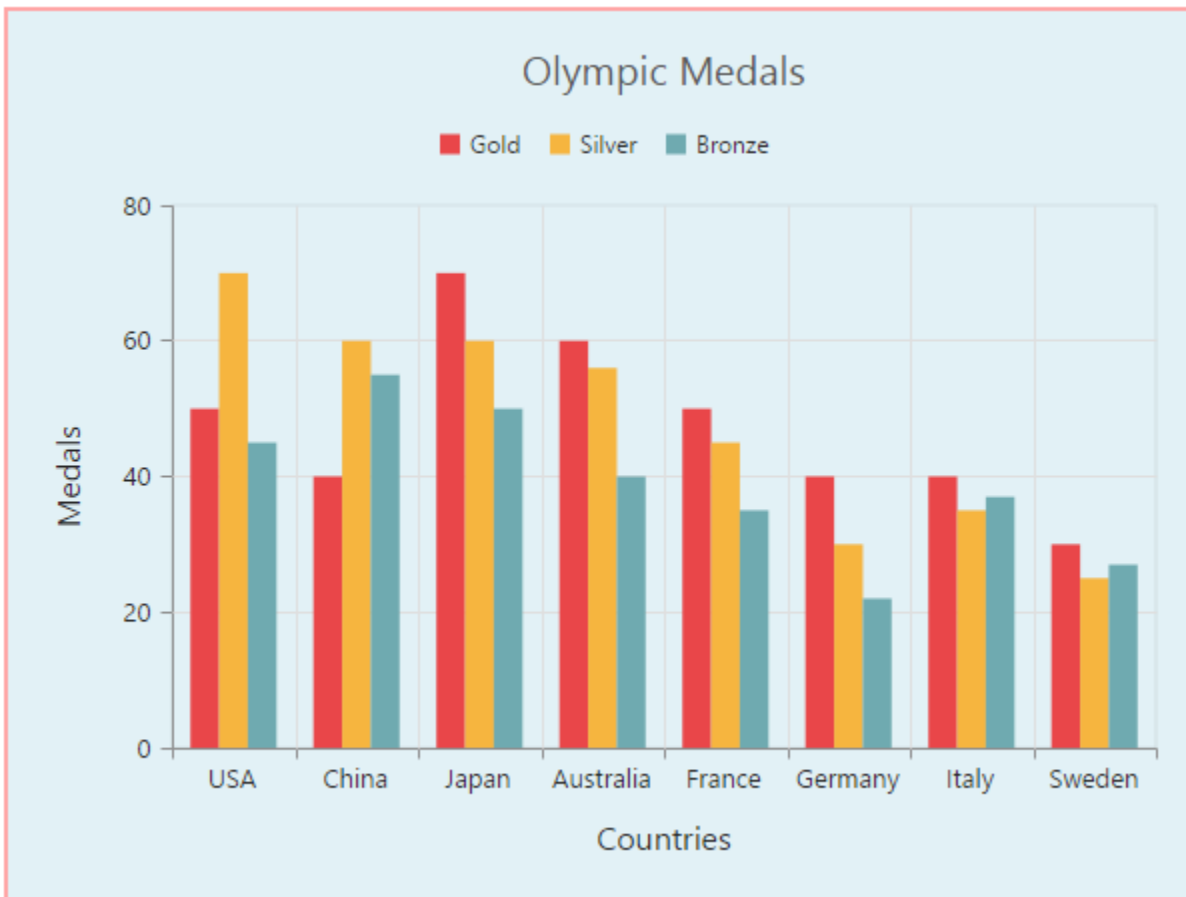
#### Customize chart background

The Chart background can be customized by using the [background](#) property of the Chart. To customize the chart border, use [border](#) option of the chart.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
});
```

```
//Customizing Chart background
background: "skyblue",
//Customize the chart border and opacity
border: {color: "#FF0000", width: 2, opacity: 0.35},
// ...
});
```

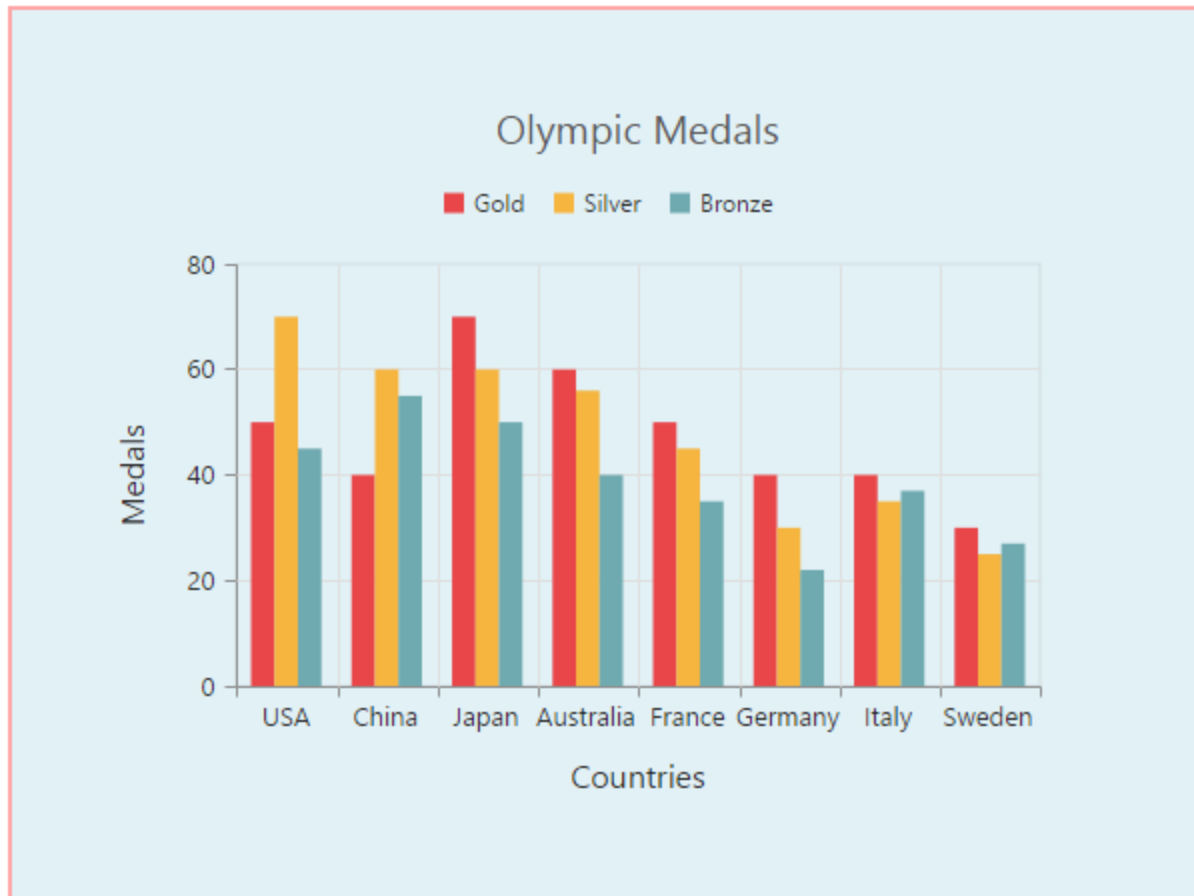


### Chart Margin

The Chart [margin](#) property is used to add the margin to the chart area at the left, right, top and bottom position.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
// ...
//Change chart margin to left, right, top and bottom
margin: { left: 40, right: 40, top: 40, bottom: 40 },
// ...
});
```



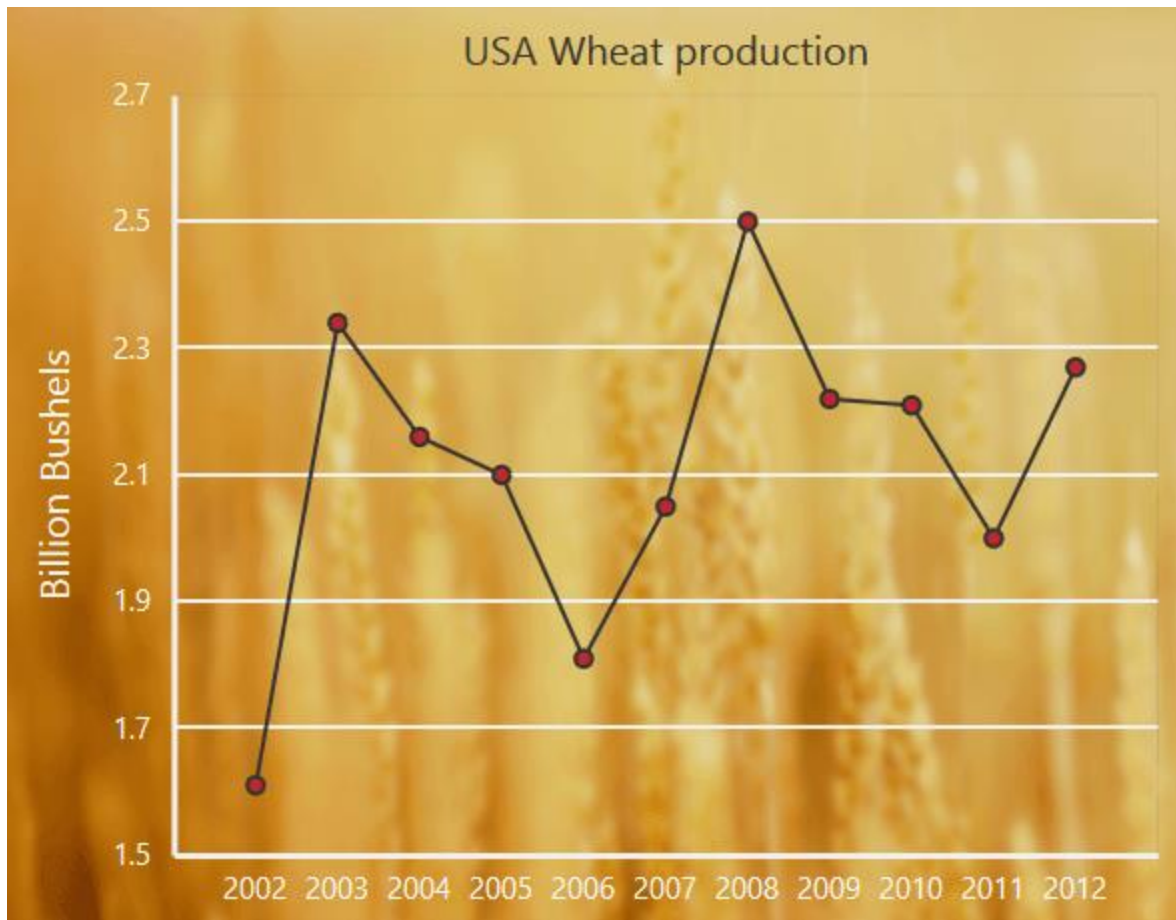
### Setting background image

Background image can be added to the chart by using the [backGroundImageUrl](#) property.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  //Setting an image as Chart background  
  backGroundImageUrl: "images/chart/wheat.png",  
  // ...  
});
```



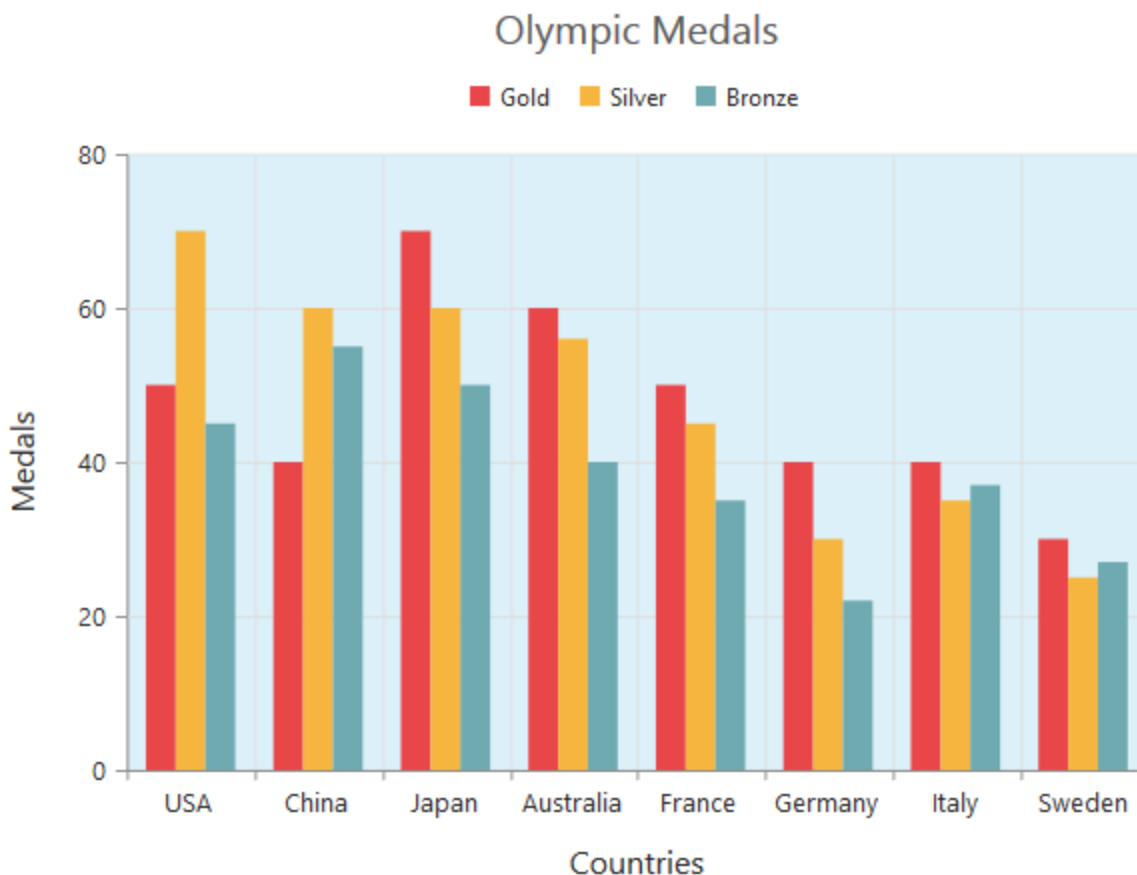


### Chart area background

The Chart area background can be customized by using the [background](#) property in the chart area.

### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {  
  // ...  
  chartArea: {  
    //Setting background for Chart area  
    background: "skyblue"  
  },  
  // ...  
});
```

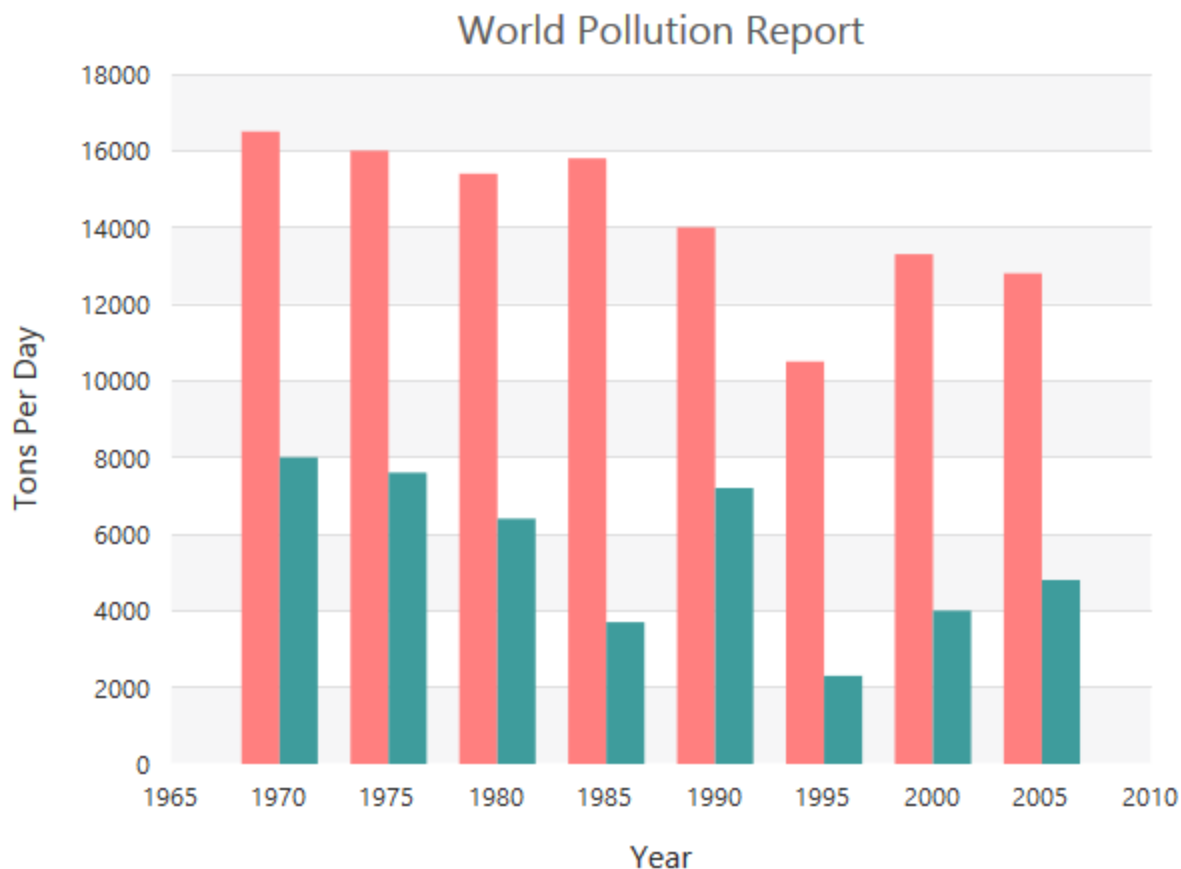


#### Customize chart area grid bands

You can provide different color for alternate grid rows and columns formed by the grid lines in the chart area by using the [alternateGridBand](#) property of the axis. The properties [odd](#) and [even](#) are used to customize the grid bands at odd and even positions respectively.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  //Creating horizontal grid bands in chart area
  primaryYAxis: {
    //Customizing horizontal grid bands at even position
    alternateGridBand: {
      even: {
        fill: "#A7A9AB",
        opacity: 0.1,
      }
    }
  },
  // ...
});
```



#### Animation

You can enable animation by using the [enableAnimation](#) property of the series. This animates the chart series on two occasions – when the chart is loaded for the first time or whenever you change the series type by using the type property.

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  // ...
  series : [{
    //Enabling animation of series
    enableAnimation: true,
    // ...
  }],
  // ...
});
```

However, you can force the chart to animate series by calling the `animate` method as illustrated in the following code example,

#### JAVASCRIPT

```
var chartSample = new ej.datavisualization.Chart($("#chartContainer"), {
  series : [{
    //Enabling animation of series
    enableAnimation: true,
```

```
// ...  
}],  
// ...  
});  
//Dynamically animating Chart  
function animateChart(){  
//Calling the animate method for dynamic animation  
$("#chartContainer").ejChart("animate");  
}
```

## Rendering Modes

Chart uses following three rendering technologies

- VML
- SVG
- HTML5 Canvas

### VML

VML is used to render chart in IE lower versions such as IE7 and IE8. VML is used by default in these two browsers, as SVG and Canvas are not supported in these browsers.

#### Limitations:

- Label rotation is not supported.
- Animation is not supported.
- Patterns are not supported.
- Zooming and panning features are not supported.

### SVG

SVG is used to render Chart by default for all browsers other than IE7 and IE8.

### Canvas

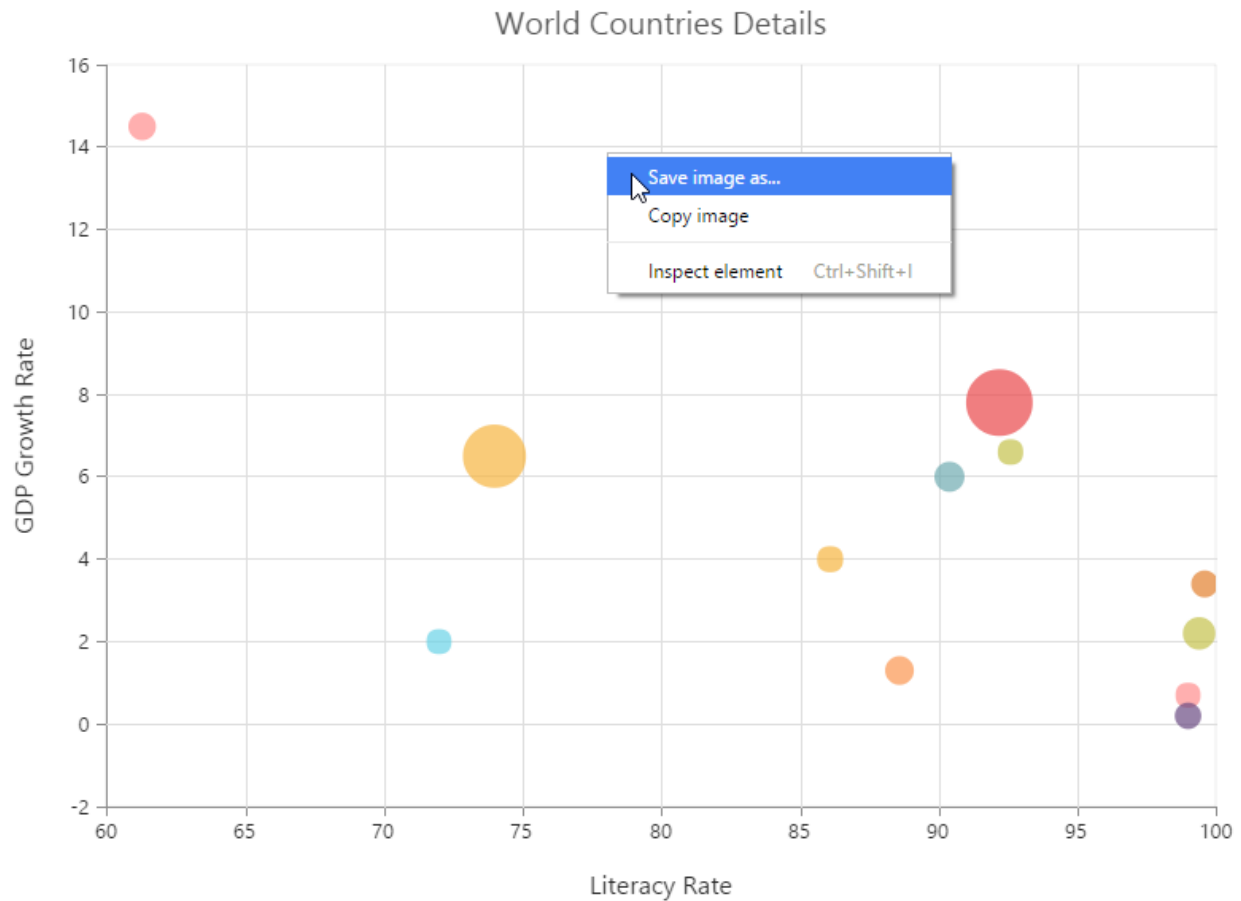
You can switch between SVG and Canvas rendering by using the [enableCanvasRendering](#) option. The canvas mode rendering is used in the following scenarios,

- Plotting large number of data points.
- Performing high frequency live updates.

The following code example shows how to enable HTML5 Canvas rendering in chart.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {  
//Rendering chart in canvas mode  
enableCanvasRendering: true  
// ...  
});
```



#### Limitations:

- Animation is not supported.
- 3D charts are not supported.

#### Real-Time Chart

Chart can be updated dynamically with the real time data. Whenever you add a point or remove a point from the dataSource, you can call the [redraw](#) method to request the chart to redraw its content.

**Note:** You can get the chart **instance** using instance method.

#### JAVASCRIPT

```
//Rendering empty Chart without data
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
  $(function () {
    var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
      series: [
        { points: [ ] }
      ],
      // ...
    });
  });
}
```

```

}
//Using set interval to update chart dynamically
window.setInterval(updateChart, 200);
//Function that updates chart dynamically
function updateChart(){
  //Creating chart instance
  var chart = $("#chartcontainer").ejChart("instance");
  if (chart.model.series[0].points.length > 10)
    chart.model.series[0].points.splice(0, 1);
  var point = chart.model.series[0].points;
  var xValue = point.length > 0 ? point[point.length - 1].x + 1 : 1;
  point[point.length] = { x: xValue, y: getRandomNum( 1000 ) }
  //Update Chart dynamically using redraw option
  //chart.redraw() can also be used here instead of redraw option
  $("#chartcontainer").ejChart("redraw");
}

```

## Printing Chart

The rendered chart can be printed directly from the browser by calling the public method [print](#). ID of the chart div element must be passed as argument to that method.

### HTML

```

<body>
<button type="button" onclick="print()" ></button>
<div id="chartcontainer"></div>
<script>
  //Render Chart
  var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  });
  function print() {
    var chartObj = $("#chartcontainer").ejChart("instance");
    chartObj.print("chartcontainer");
  }
</script>
</body>

```

This print method can be called by performing any action on the web page. For example, by clicking a button. While calling the print method in chart, print preview will be displayed in the browser.

Print

Total: **1 page**

Save

Cancel

Destination

Save as PDF

Change...

Pages

☒ All

☐ e.g. 1-5, 8, 11-13

Layout

Portrait

+

More settings

Sales Comparison

| Manager | Sales  |
|---------|--------|
| John    | 10,000 |
| Jake    | 12,000 |
| Peter   | 18,000 |
| James   | 11,000 |
| Mary    | 9,500  |

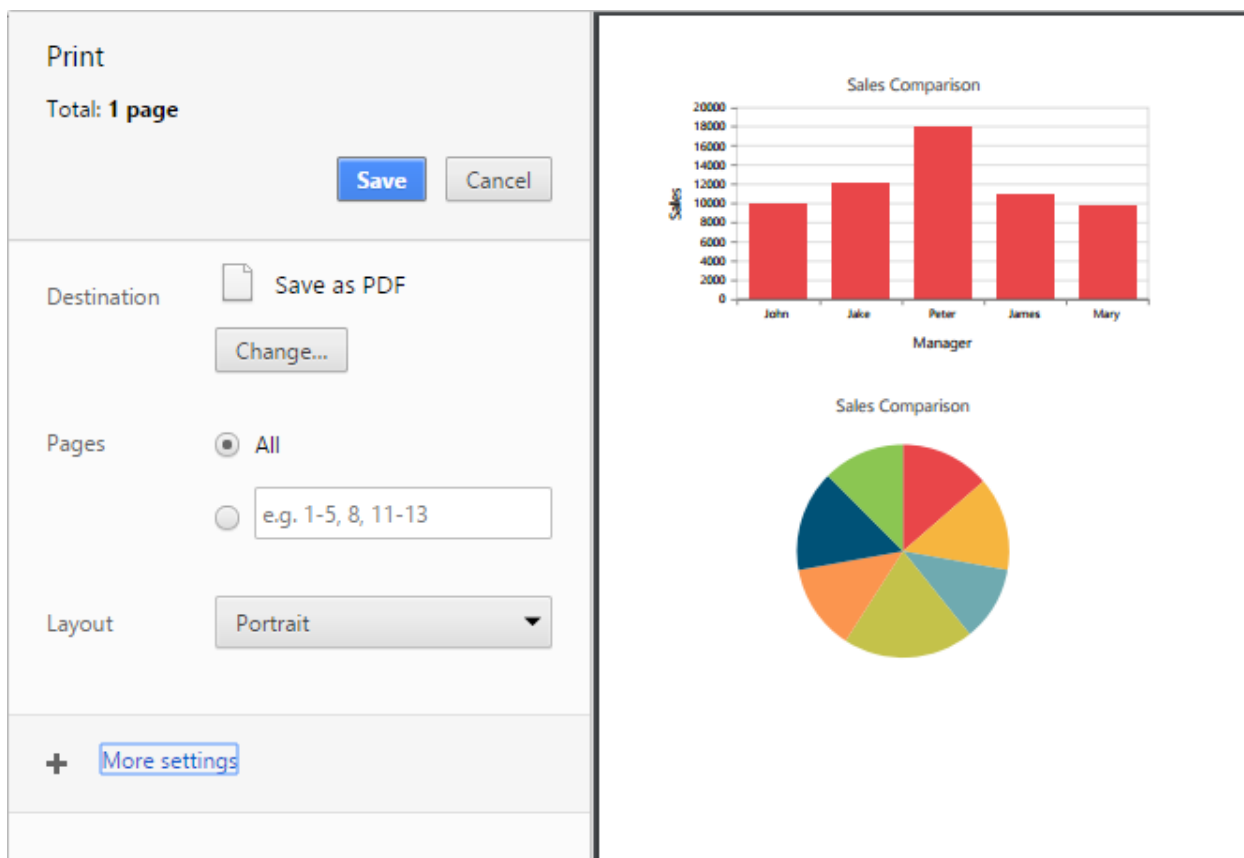
### Printing Multiple chart

Multiple charts in a web page can be printed together. For printing multiple charts, ID of the chart div elements have to be passed as shown in the below code

### JAVASCRIPT

```
//Render Chart1
var chartsample = new ej.datavisualization.Chart($("#container1"), {
})
//Render Chart2
var chartsample = new ej.datavisualization.Chart($("#container2"), {
})
//Print multiple charts
function print() {
var chartObj = $("#container1").ejChart("instance");
chartObj.print("container1", "container2");
}
```

The Print preview of multiple Charts is shown below



### Page Setup

Some of print options are not configurable through JavaScript code. You need to customize layout, paper size, margins options through browser's page setup dialog. Please find the following guidelines link to browser page setup.

- [Chrome](#)
- [Firefox](#)
- [Safari](#)
- [IE](#)

### 3D Chart

Essential 3D Chart for JavaScript allows you to view 8 chart types in 3D view such as [Bar](#), [StackingBar](#), [StackingBar100](#), [Column](#), [Stacked Column](#), [100% Stacked Column](#), [Pie](#), [Doughnut](#).

#### 3D Column Chart

For rendering a 3D Column Chart, specify the series [type](#) as “**column**” in the chart series and set [enable3D](#) option as **true** in the chart.

#### JAVASCRIPT

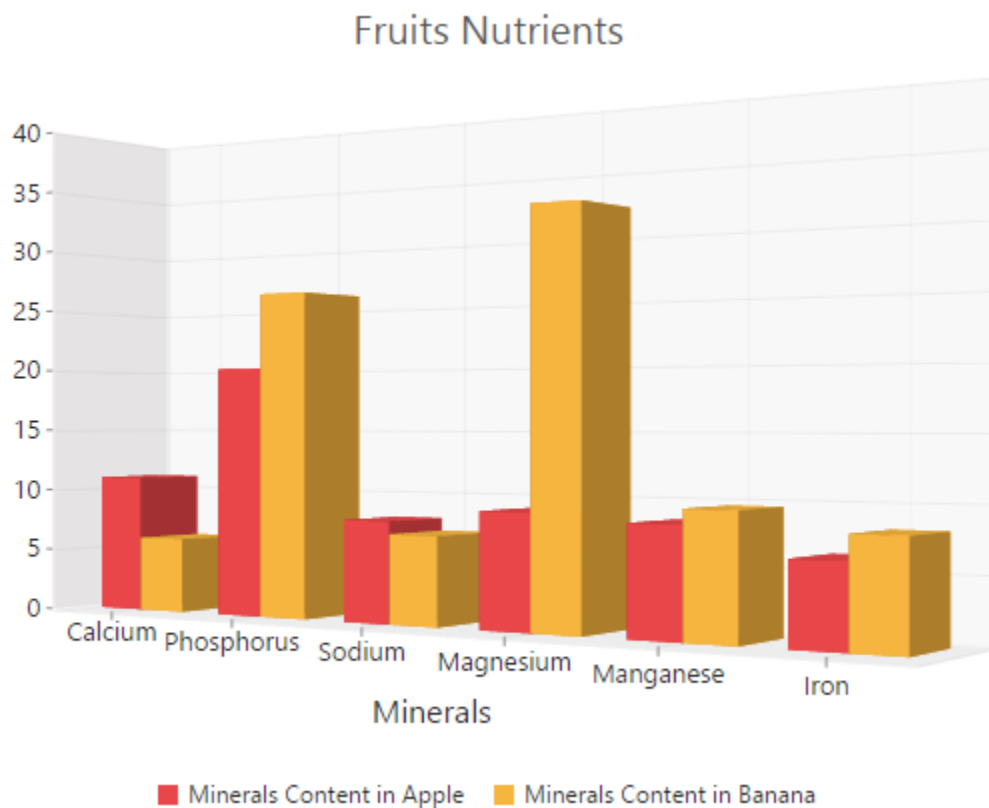
```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
$(function () {
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {

```



```
// ...
series: [{
  //Set chart type to series
  type: 'column',
  // ...
}],
// Enable 3D Chart
enable3D: true,
// ...
});
});
}
```



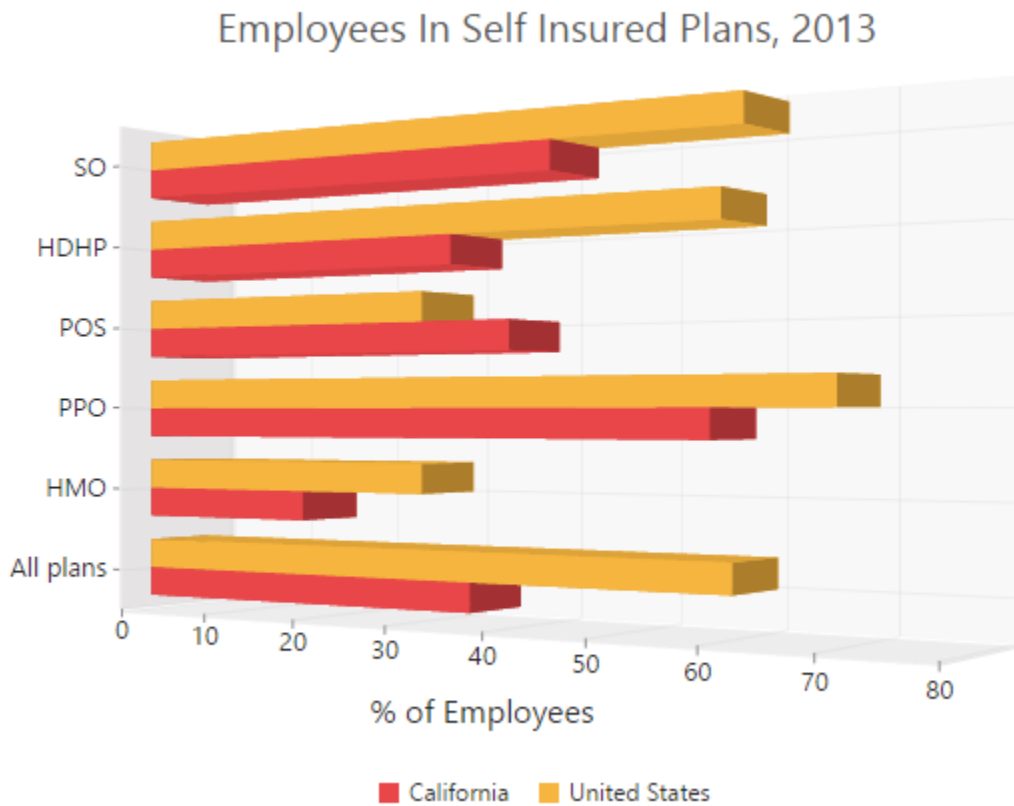
### 3D Bar Chart

You can create a 3D Bar Chart by setting the series [type](#) as “**bar**” in the chart series and enable [enable3D](#) option in the chart.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  series: [{
    //Set chart type to series
    type: 'bar',
    // ...
  }],
  // Enable 3D Chart
});
```

```
enable3D: true,
// ...
});
```

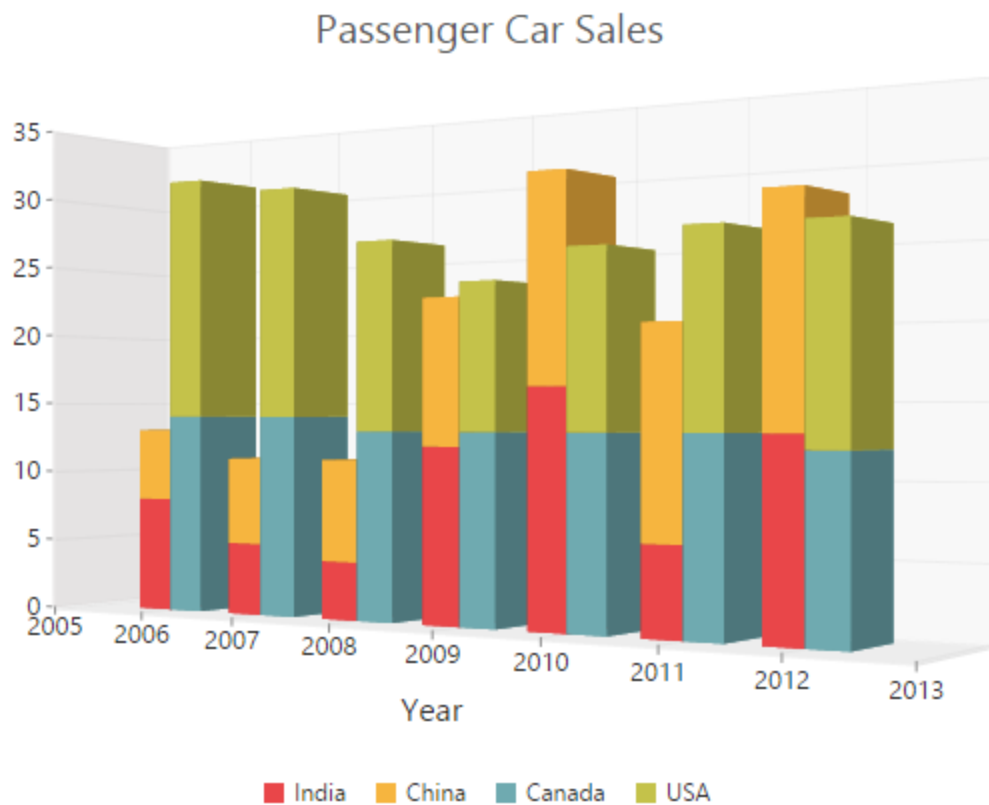


### 3D Stacked Column Chart

Stacking Column 3DChart is rendered by specifying the series [type](#) as “**stackingColumn**” in the chart series and enable [enable3D](#) option in the chart.

### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
// ...
series: [{
//Set chart type to series1
type: 'stackingColumn',
// ...
},{
//Set chart type to series2
type: 'stackingColumn',
// ...
}],
// Enable 3D Chart
enable3D: true,
// ...
});
```

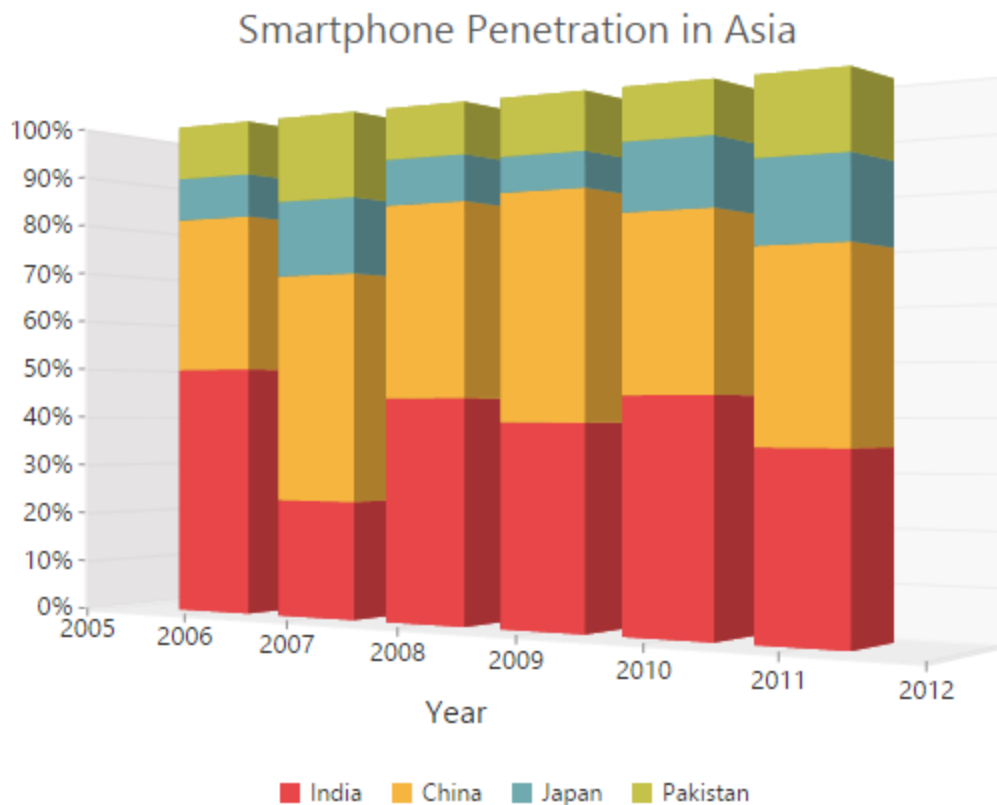


### 3D 100% Stacked Column Chart

100% Stacking Column 3DChart is rendered by specifying the series [type](#) as “**stackingColumn100**” in the chart series and enable [enable3D](#) option in the chart.

### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  series: [{
    //Set chart type to series1
    type: 'stackingColumn100',
    // ...
  }, {
    //Set chart type to series2
    type: 'stackingColumn100',
    // ...
  }],
  // Enable 3D Chart
  enable3D: true,
  // ...
});
```

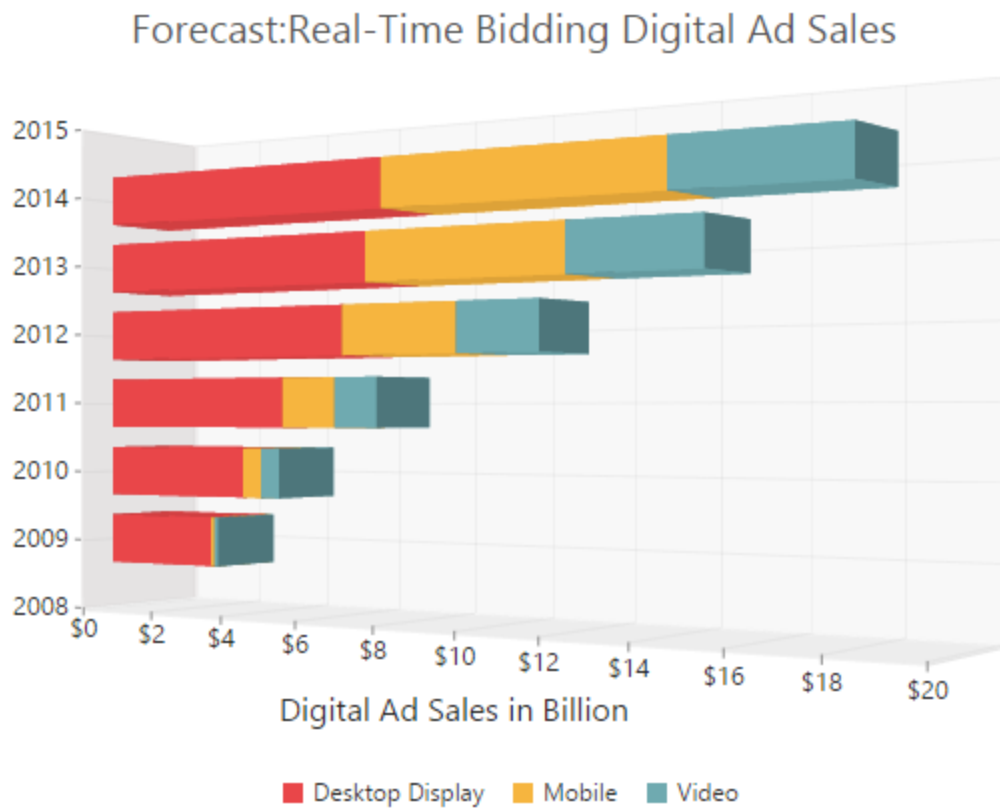


### 3D Stacked Bar Chart

To create Stacking Bar 3DChart, set the series [type](#) as “**stackingBar**” in the chart series and enable [enable3D](#) option in the chart.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  series: [{
    //Set chart type to series1
    type: 'stackingBar',
    // ...
  }, {
    //Set chart type to series2
    type: 'stackingBar',
    // ...
  }],
  // Enable 3D Chart
  enable3D: true,
  // ...
});
```

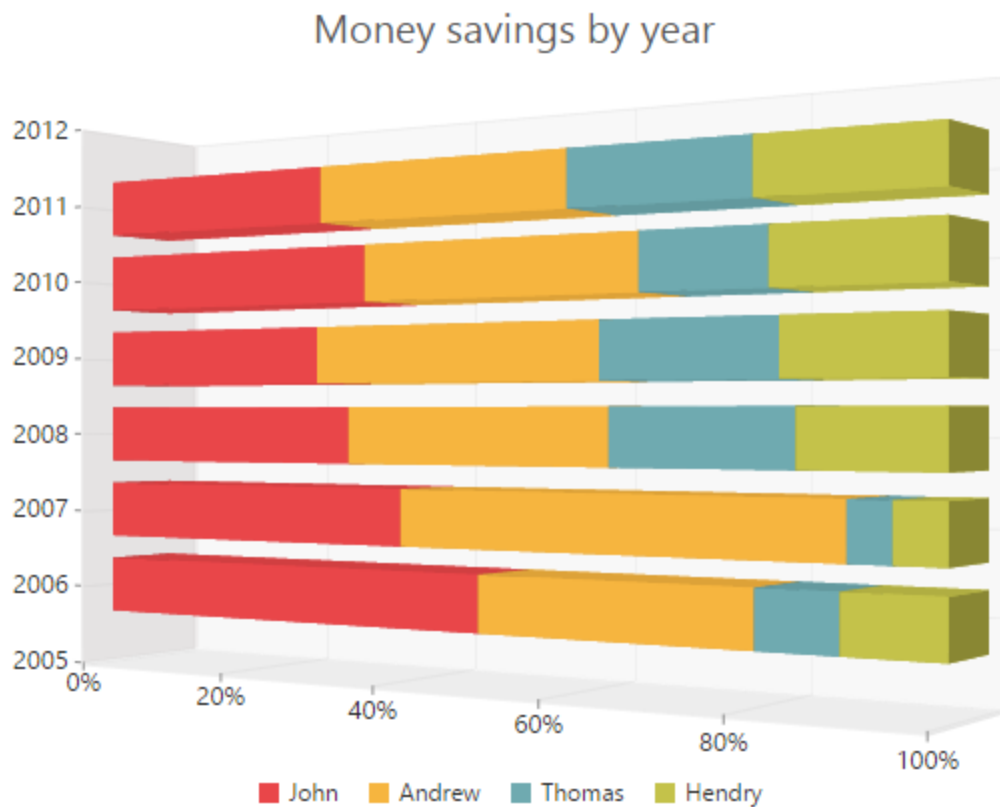


### 3D 100% Stacked Bar Chart

You can create 100% Stacking Bar 3DChart by setting the series [type](#) as “**stackingbar100**” in the chart series and enable [enable3D](#) option in chart.

### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  series: [{
    //Set chart type to series1
    type: 'stackingBar100',
    // ...
  }, {
    //Set chart type to series2
    type: 'stackingBar100',
    // ...
  }],
  // Enable 3D Chart
  enable3D: true,
  // ...
});
```

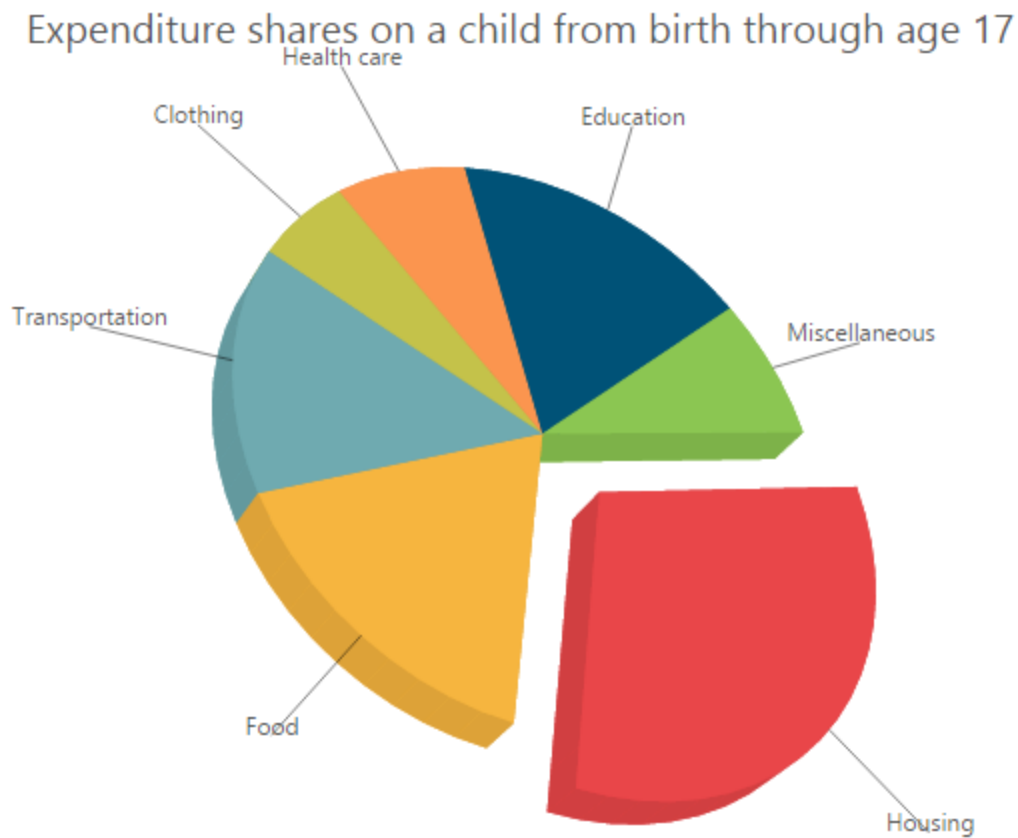


### 3D Pie Chart

To render the Pie Chart in 3D view, enable the **enable3D** option in the chart and set the series **type** as **"pie"** in the chart series.

### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
  // ...
  series: [{
    //Set chart type to series
    type: 'pie',
    // ...
  }],
  // Enable 3D Chart
  enable3D: true,
  // ...
});
```

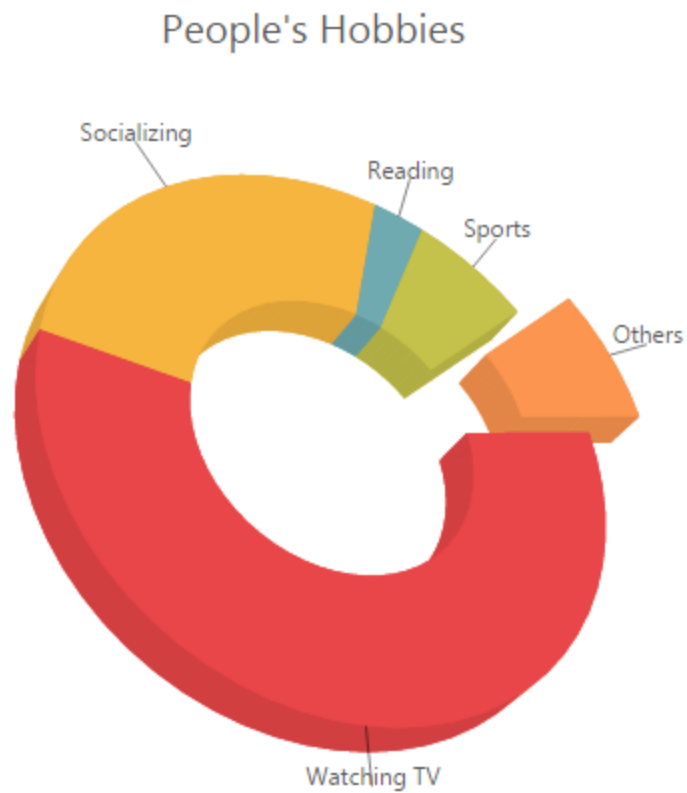


### 3D Doughnut Chart

To render the Doughnut Chart in 3D view, enable the **enable3D** option in the chart and set the series [type](#) as “**doughnut**” in the chart series.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {  
  // ...  
  series: [{  
    //Set chart type to series  
    type: 'doughnut',  
    // ...  
  }],  
  // Enable 3D Chart  
  enable3D: true,  
  // ...  
});
```



Configure 3D Chart

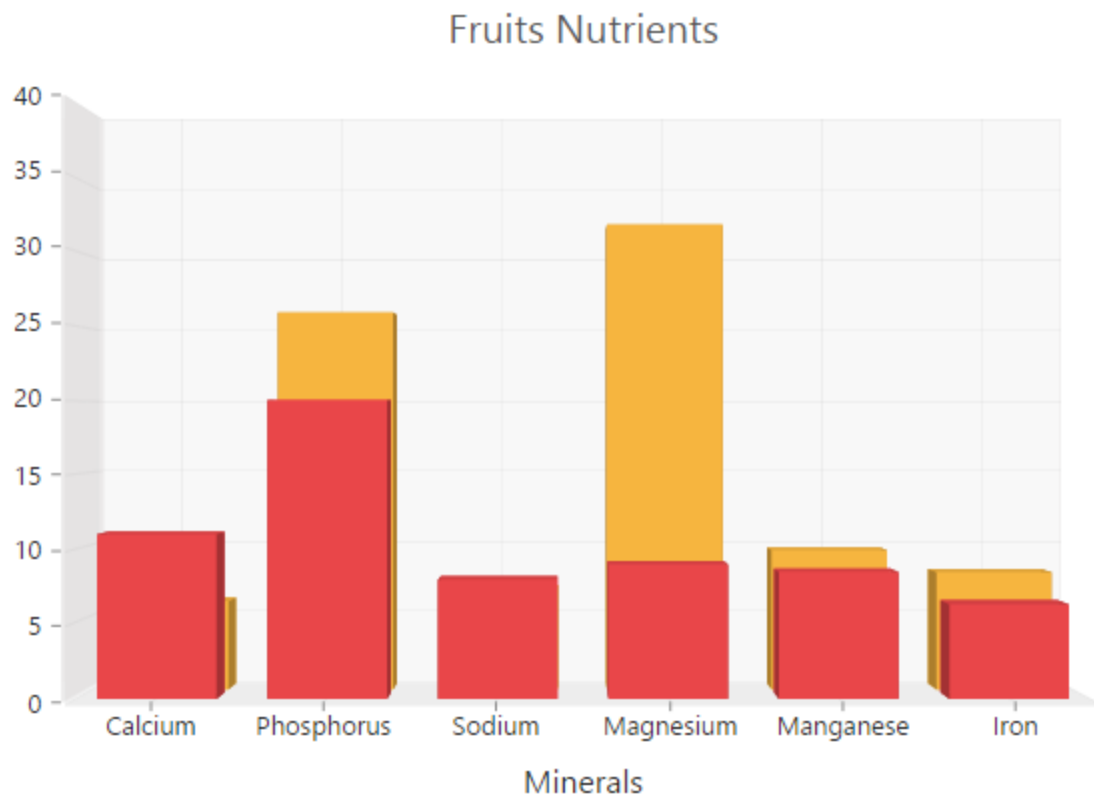
*3D View*

To render the EjChart in 3D view, set the [enable3D](#) option as *true* in the chart.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {  
  // ...  
  //Enable 3DChart  
  enable3D: true,  
  // ...  
});
```



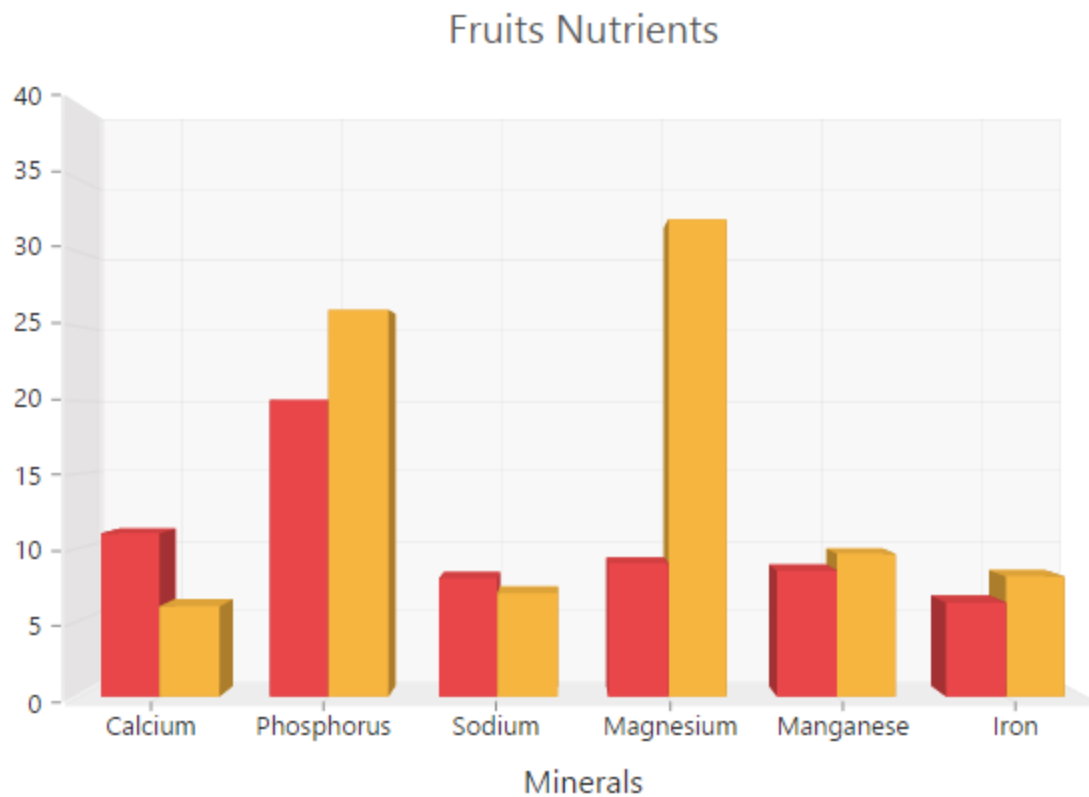


*Placing Bar / Column kind of series side-by-side*

The [sideBySideSeriesPlacement](#) defines the appearance of the different sets of data on the 3D Chart. When this property is enabled, the data is displayed side by side, otherwise it is displayed along the depth of the axis.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {  
  // ...  
  //Enable SideBySideSeriesPlacement for 3DChart  
  sideBySideSeriesPlacement: true,  
  // ...  
});
```

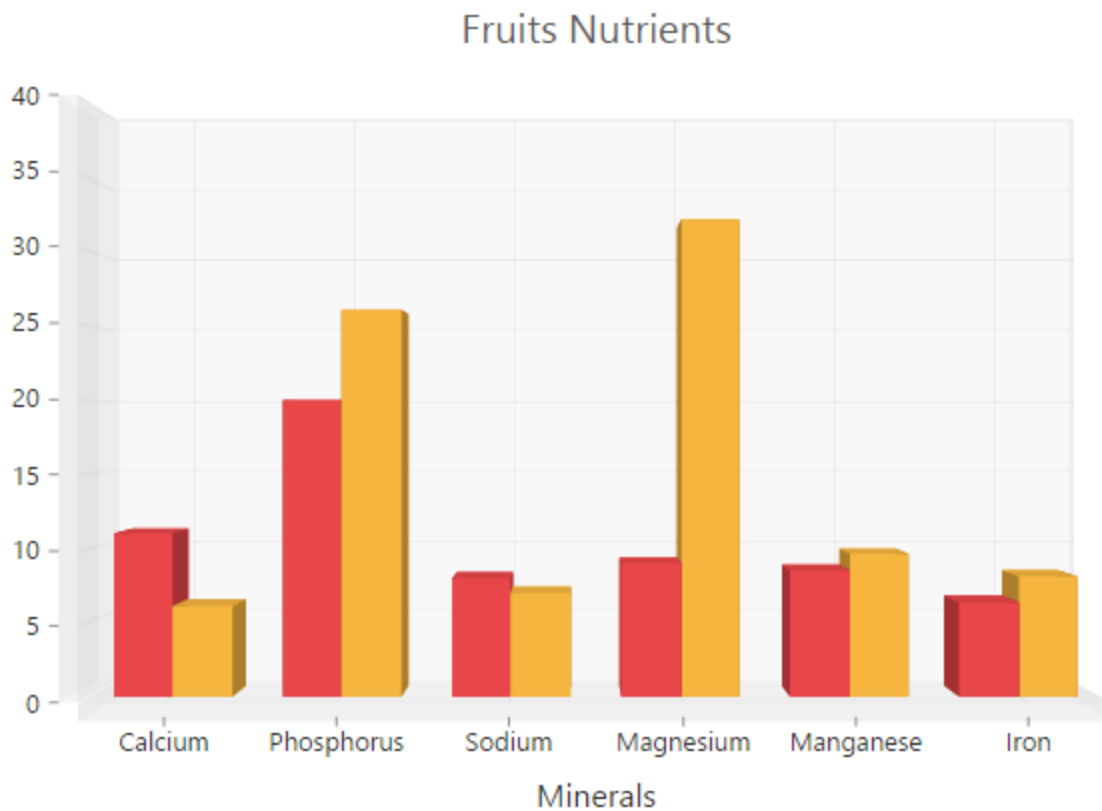


#### Setting Axis Wall Size

In 3DChart, Cartesian axes lines are represented as walls and it defines the width of the 3D wall. 3D Pie and Doughnut Chart does not support [wallSize](#) because they don't have axes.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {  
  // ...  
  //Change 3DChart axis wall size  
  wallSize: 10,  
  // ...  
});
```

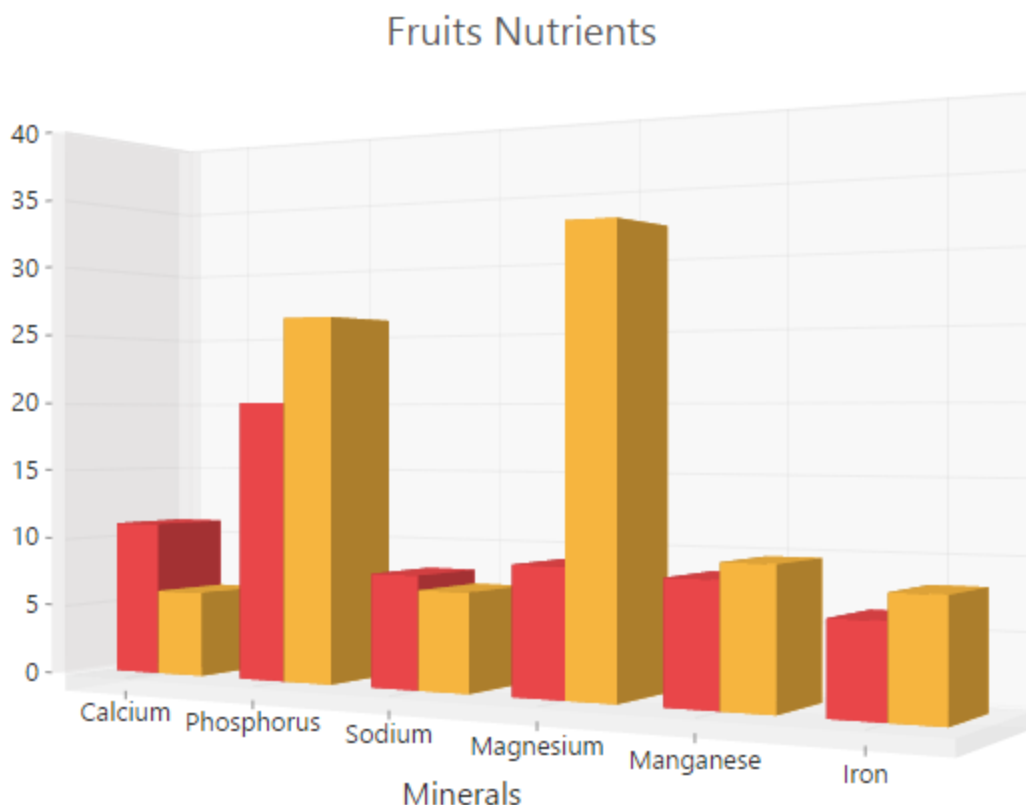


#### 3D Depth

By using the [depth](#) property, you can view the 3D Chart from the front view of the series to the background wall.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {  
  // ...  
  //Change 3DChart depth value  
  depth: 120,  
  // ...  
});
```

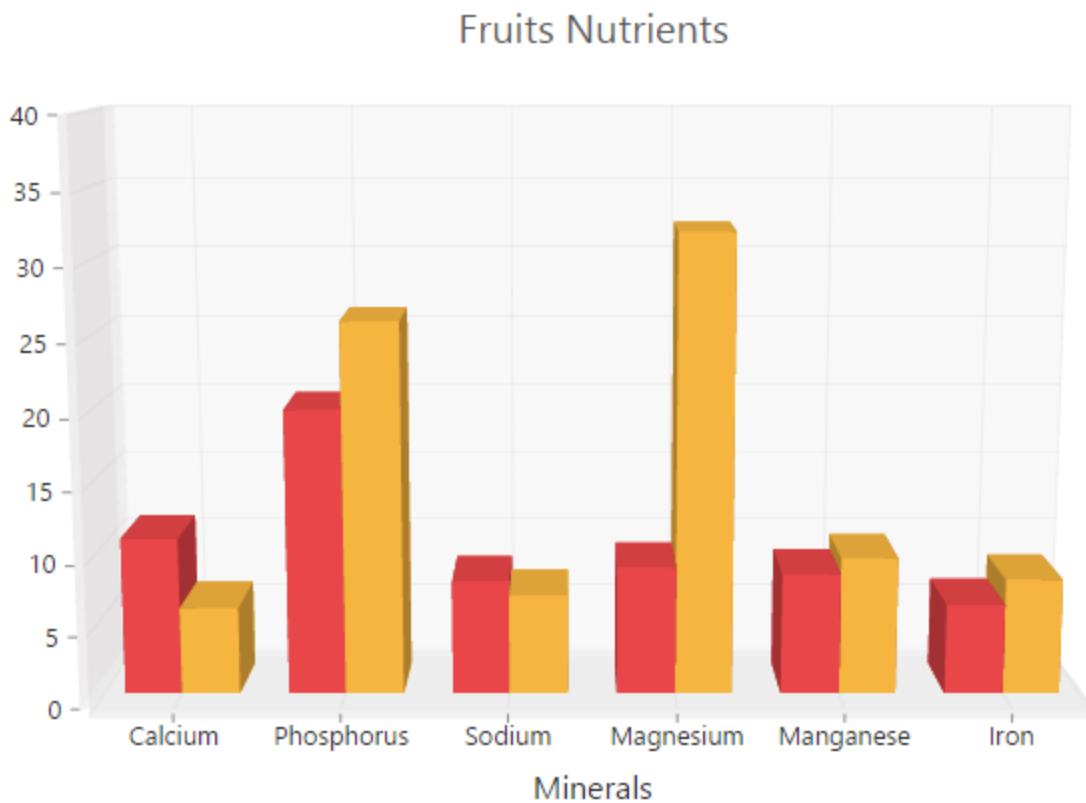


#### Rotating and Tilting 3D Chart

To spin the 3D Chart on mouse dragging, enable [enableRotation](#) option in the chart. The [tilt](#) property specifies the angle of the slope of the 3D Chart. The positive and negative values are declared to the side where the slope is present. The [rotation](#) option is used to rotate the 3D chart towards left or right side of the chart. The direction of the chart depends upon the positive and negative values of the angle.

#### JAVASCRIPT

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {  
  // ...  
  //Change 3DChart tilt value  
  tilt: 10,  
  //Enable 3DChart rotation on mouse dragging  
  enableRotation: true,  
  //Change 3DChart rotation on chart load  
  rotation: 40,  
  // ...  
});
```

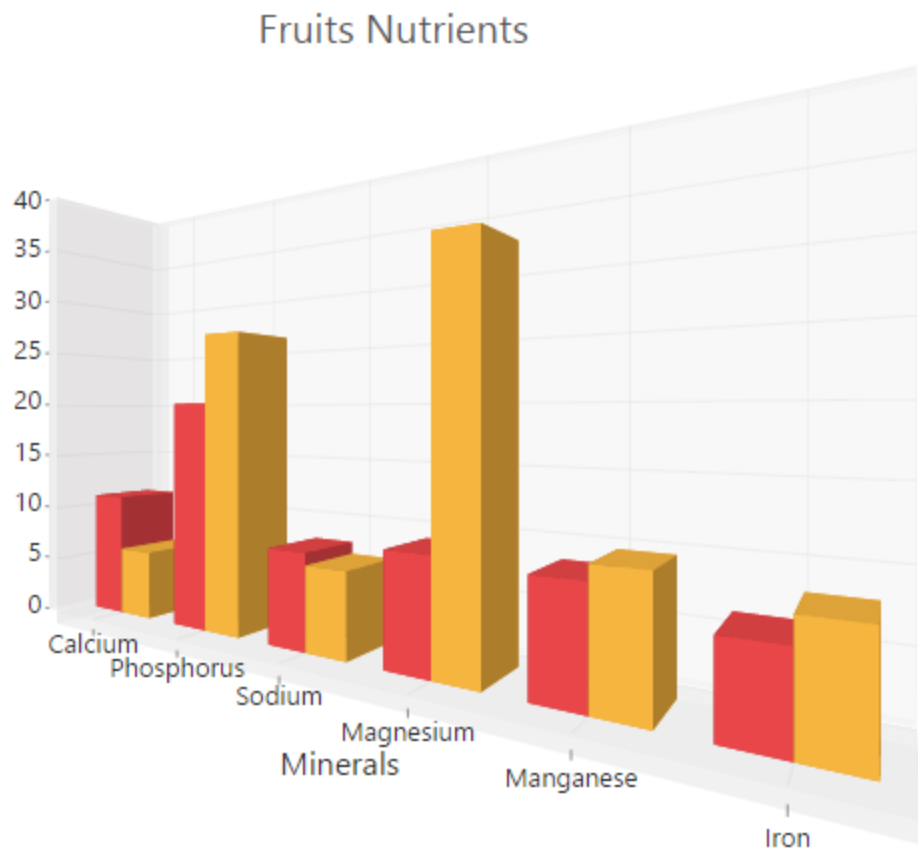


#### *PerspectiveAngle*

The [perspectiveAngle](#) specifies the appearance of the height, width, depth and wall of the 3D Chart. When the perspectiveAngle is decreased, the 3D object appears very close to the viewer. But when it is increased, the 3D object appears far away from the viewer.

#### **JAVASCRIPT**

```
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {  
  // ...  
  //Change 3DChart perspective angle  
  perspectiveAngle: 150,  
  // ...  
});
```

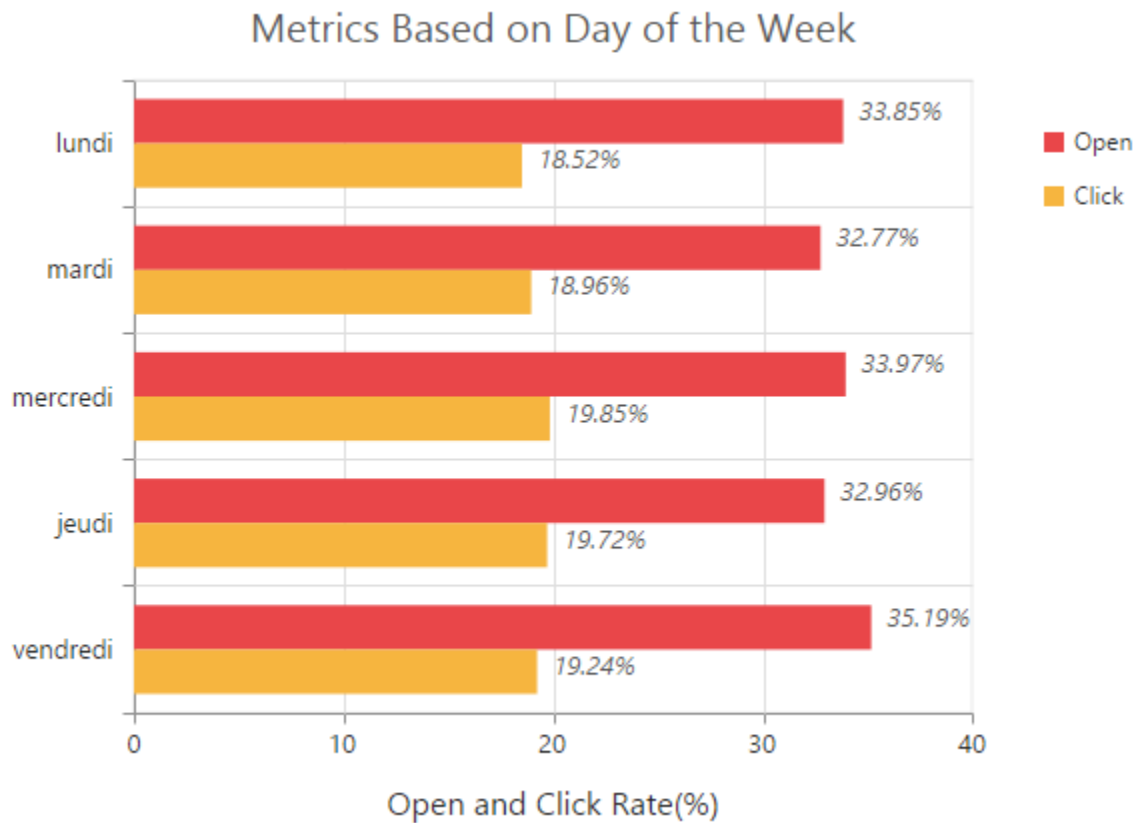


## Localization

EjChart supports localization for its axis labels and tooltip. To render the chart with specific culture you have to refer the corresponding **globalize** culture script and need to specify the culture name in [locale](#) property of chart.

## HTML

```
<head>
<!--Refer french globalize culture script-->
<script src="../../scripts/cultures/globalize.culture.fr-FR.min.js"></script>
</head>
<body>
<div id="chartcontainer"></div>
<script>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ChartComponent {
$(function () {
var chartsample = new ej.datavisualization.Chart($("#chartcontainer"), {
// ...
//Render chart in french locale
locale: 'fr-FR',
});
});
</script>
</body>
```



## Public Methods

### *animate(options)*

Animates the series and/or indicators in Chart. When parameter is not passed to this method, then all the series and indicators present in Chart are animated.

Following are the parameters that you can pass to this method.

Returns: void

| Parameters | Type   | Description  |
|------------|--------|--|
| options    | object | <p>If an array collection is passed as parameter, series and indicator objects passed in array collection are animated.</p> <p>Example{% highlight js %}var chartObj =<br/>\$("#container").data("ejChart");//animating series<br/>arraychartObj.animate(chartObj.model.series);{% endhighlight %}</p> <p>If a series or indicator object is passed to this method, then the specific series or indicator is animated.</p> <p>Example,{% highlight js %}var chartObj =</p> |

| Parameters | Type | Description   |
|------------|------|---|
|            |      | <code>\$("#container").data("ejChart");//animating a specific indicatorchartObj.animate(chartObj.model.indicators[0]);{% endhighlight %}</code> |

*print()*

Prints the rendered chart.

Returns: void

Example

**JS**

```
// Print Chart
var chartObj = $("#container").data("ejChart");
chartObj.print();
```

If you wish to print multiple charts on a same page, then you need to pass the ID of those elements as arguments to the print method.

```
<div id="container1"></div>
```

```
<div id="container2"></div>
```

**JS**

```
// Print Chart
var chartObj = $("#container1").data("ejChart");
chartObj.print("container1", "container2");
```

*export(type, URL, exportMultipleChart)*

Exports chart as an image or to an excel file. Chart can be exported as an image only when exportCanvasRendering option is set to true.

Following are the parameters that you can pass to this method,

Returns: object

| Parameters | Type   | Description  |
|------------|--------|--|
| type       | string | Type of the export operation to be performed. Following are the two export types that are supported now,<br>1. 'image' 2. 'excel'<br>Example{% highlight js %}var chartObj =<br>\$("#container").data("ejChart");chartObj.export("image");{% endhighlight %}             |
| URL        | string | URL of the service, where the chart will be exported to excel.<br>Example,{% highlight js %}var chartObj =<br>\$("#container").data("ejChart");chartObj.export("excel",<br>'http://js.syncfusion.com/ExportingServices/api/JSChartExport/ExcelExport'){% endhighlight %} |



| Parameters          | Type    | Description   |
|---------------------|---------|---|
| exportMultipleChart | boolean | <p>When this parameter is true, all the chart objects initialized to the same document are exported to a single excel file. This is an optional parameter. By default, it is false.</p> <p>Example,{% highlight js %}var chartObj = \$("#container").data("ejChart");chartObj.export("excel", 'http://js.syncfusion.com/ExportingServices/api/JSChartExport/ExcelExport', true){% endhighlight %}</p> |

### redraw()

Redraws the entire chart. You can call this method whenever you update, add or remove points from the data source or whenever you want to refresh the UI.

Returns: void

### Example

#### JS

```
// Redraw Chart
var chartObj = $("#container").data("ejChart");
chartObj.redraw();
```

#### JS

```
$("#container").ejChart("redraw");
```

## Checkbox

### Overview

**CheckBox** component is an input component with trendy look and appearance used to input multiple options from the user.

### Key Features

- **Trendy Look** : Rich appearance with theme Support
- **RTL** : Supports for right to left alignment
- **Tri-State**: Supports **CheckBox** with three states.
- **Easy Customization**: The style and appearance of CheckBox component could be easily configured.
- **Themes**: Supports 17 built-in themes (6 – flat and 6 – gradient effects), and also support custom skin options to set user-defined themes.
- **Responsive and Touch support**: Fits in all range of devices and supports touch devices

### Getting Started

This section discloses the details on how to render and configure a CheckBox component in a TypeScript application.

### Create your first CheckBox

1. Create a TypeScript application and refer the dependent modules, script and CSS with the help of given getting started document.
2. In the index.HTML file, add the input element for rendering CheckBox component as given below.

#### HTML

```
<div>
Hobbies <br /><br />
<table>
<tr>
<td>
<ej-checkbox id="Checkbox1"></ej-checkbox>
<label for="Checkbox1">Music</label>
</td>
<td>
<ej-checkbox id="Checkbox2"> </ej-checkbox>
<label for="Checkbox2">Sports</label>
</td>
<td>
<ej-checkbox id="Checkbox3"> </ej-checkbox>
<label for="Checkbox3">Bike Riding</label>
</td>
</tr>
</table><br /><br />
Favorite Search Engines<br /><br />
<table>
<tr>
<td>
<ej-checkbox id="Checkbox4"> </ej-checkbox>
<label for="Checkbox4">Google</label>
</td>
<td>
<ej-checkbox id="Checkbox5"> </ej-checkbox>
<label for="Checkbox5">Yahoo</label>
</td>
<td>
<ej-checkbox id="Checkbox6"> </ej-checkbox>
<label for="Checkbox6">Bing</label>
</td>
</tr>
</table>
</div>
```

3. Create a TypeScript file named "app.ts" file and refer the required definition files as given below.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

4. Now, initialize the CheckBox component by using ej.CheckBox method.

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module AppComponent {
  $(function () {
    new ej.CheckBox($("#Checkbox1"), {
      value: "Music", name: "Hobbies", size: "small",
      enableTriState: true, checked: true
    });
    new ej.CheckBox($("#Checkbox2"), {
      value: "sports", name: "Hobbies", size: "small",
      enableTriState: true, checkState: "indeterminate"
    });
    new ej.CheckBox($("#Checkbox3"), {
      value: "Bike riding", name: "Hobbies", size: "small",
      enableTriState: true
    });
    new ej.CheckBox($("#Checkbox4"), {
      value: "Google", name: "SearchEngines", size: "medium",
      enableTriState: true, checked: true
    });
    new ej.CheckBox($("#Checkbox5"), {
      value: "Yahoo", name: "SearchEngines", size: "medium",
      enableTriState: true, checkState: "indeterminate"
    });
    new ej.CheckBox($("#Checkbox6"), {
      value: "Bing", name: "SearchEngines", size: "medium",
      enableTriState: true
    });
  });
}
```

### Run the application

To run the application, navigate the project folder and open command prompt window and execute the following command.

### JS

```
tsc
```

This command compiles the app.ts file to generate a JS file named app.js file.

Refer the app.js file in index.html and browse the html file to see the following output.

## Hobbies

☒ Music ☐ Sports ☐ Bike Riding

## Favorite Search Engines

☒ Google ☐ Yahoo ☐ Bing

## Easy customization

### Checked status

Using **checked** property, you can set the state of Checkbox. When checked property is true then tick mark is displayed and Checkbox is in checked state. When it is false, the tick mark is not displayed and Checkbox is in unchecked state. When you want to use this **checked** property, then checkbox should be in non Tri-state and **enableTriState** property should be false.

The following steps explains you the details about rendering the Checkbox with above mentioned checked options, when the checkbox is in non tri-state.

In the **HTML** page, add the following input elements to configure Checkbox widget.

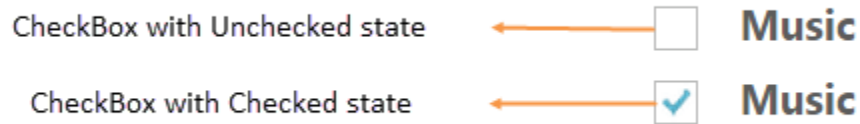
### HTML

```
<div class="align">
<input type="checkbox" class="nodetext" id="checkbox_nonChecked" />
<label for="checkbox_nonChecked" class="clslab">Music</label>
<br />
<input type="checkbox" class="nodetext" id="checkbox_checked" />
<label for="checkbox_checked" class="clslab">Music</label>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CheckBoxComponent {
$(function () {
var checkbox = new ej.CheckBox($("#checkbox_nonChecked"), {
checked: false
});
//Enables the checked status
var checkbox1 = new ej.CheckBox($("#checkbox_checked"), {
checked: true
});
});
}
```

Execute the above code to render the following output.



### Enable Tri-State

Sometimes, it is essential for you to represent the answer in partially true state. To represent the partially true types, an indeterminate state option is present. The state between checked and unchecked state is called indeterminate state. For example, a **Checkbox** presented to select files to send via [FTP](#) can use a [tree view](#) so that files can be selected one at a time, or by folder. When only some of the files in a folder are selected, then the checkbox for that folder could be in indeterminate state.

When you enable Tri-state, then the **Checkbox** includes the indeterminate state. The Checkbox has three states. **enableTriState** property specifies to enable or disable the Tri-State option for Checkbox.

The following steps explain you the details about rendering the Checkbox with Tri-state options.

In the **HTML** page, add the following input elements to configure Checkbox widget.

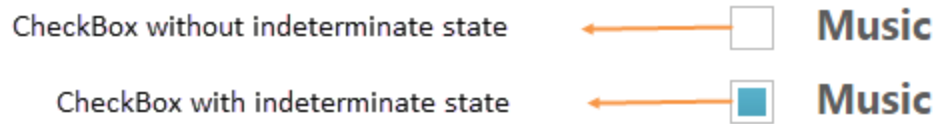
### HTML

```
<div class="align">
<input type="checkbox" class="nodetext" id="checkbox_nonTriState" />
<label for="checkbox_nonTriState" class="clslab">Music</label>
<br />
<input type="checkbox" class="nodetext" id="checkbox_triState" />
<label for="checkbox_triState" class="clslab">Music</label>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CheckBoxComponent {
$(function () {
var checkbox = new ej.CheckBox($("#checkbox_nonTriState"), {
enableTriState: false
});
var checkbox1 = new ej.CheckBox($("#checkbox_triState"), {
enableTriState: true,
checkState:"indeterminate"
});
});
}
```

Execute the above code to render the following output.



### Check State

You require an option to set indeterminate state for Checkbox. By using Checkbox property, you can set any state that is illustrated in following table. Before using this property, enable the Tri-state for Checkbox. **enableTriState** property is set true.

List of check states

| Check States  | Description                              |
|---------------|--|
| check         | Check box will be in checked state       |
| uncheck       | Check box will be in un-checked state    |
| indeterminate | Check box will be in indeterminate state |

The following steps explains you the details about rendering the **Checkbox** with specified checked state, when the checkbox is in tri-state.

In the **HTML** page, add the following input elements to configure **Checkbox** widget.

### HTML

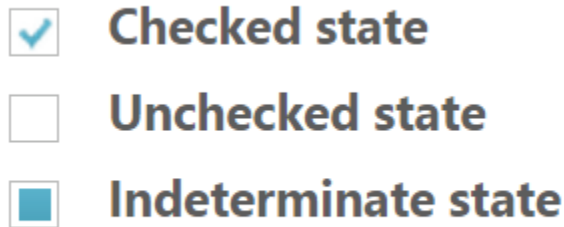
```
<div class="align">
<input type="checkbox" class="nodetext" id="check" />
<label for="check" class="clslab">Checked state</label>
<br />
<input type="checkbox" class="nodetext" id="uncheck" />
<label for="uncheck" class="clslab">Unchecked state</label>
<br />
<input type="checkbox" class="nodetext" id="indeterminate" />
<label for="indeterminate" class="clslab">Indeterminate state</label>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CheckBoxComponent {
$(function () {
var checkbox = new ej.CheckBox($("#check"), {
enableTriState: true,
checkState: "check"
});
var checkbox1 = new ej.CheckBox($("#uncheck"), {
enableTriState: true,
checkState: "uncheck"
});
});
}
```

```
});
var checkbox2 = new ej.CheckBox($("#indeterminate"), {
  enableTriState: true,
  checkState: "indeterminate"
});
});
}
```

Execute the above code to render the following output.



### Checkbox Size

You can render **Checkbox** in different sizes. The following table contains some predefined size option for rendering a **Checkbox** in easiest way. Each size option has different height and width. Mainly it avoids the complexity in rendering **Checkbox** with complex **CSS** class.

List of checkbox size:

| CheckBox size | Description   |
|---------------|---|
| small         | Creates checkbox with Built-in small size height, width specified.  |
| medium        | Creates checkbox with Built-in medium size height, width specified. |

The following steps explains you the details about rendering the **Checkbox** with different size.

In the **HTML** page, add the following input elements to configure **Checkbox** widget.

### HTML

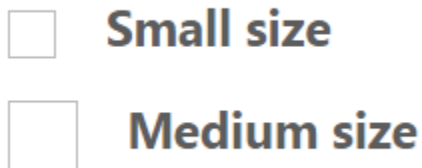
```
<div class="align">
  <input type="checkbox" class="nodetext" id="checkbox_small" />
  <label for="checkbox_small" class="clslab">Small size</label>
<br />
  <input type="checkbox" class="nodetext" id="checkbox_medium" />
  <label for="checkbox_medium" class="clslab">Medium size</label>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CheckBoxComponent {
```

```
$(function () {  
  var checkbox = new ej.CheckBox($("#checkbox_small"), {  
    size: "small"  
  });  
  var checkbox = new ej.CheckBox($("#checkbox_medium"), {  
    size: "medium"  
  });  
});
```

Execute the above code to render the following output.



### Text

It specifies the text content for **Checkbox**. In previous programs, separate label for each Checkbox is created. You can also set the text for checkbox using **text** property. Therefore, it is not essential to add label tag for each checkbox in **HTML** code.

The following steps explain you the details about rendering the Checkbox with text content and without writing label tag in **HTML** code

In the **HTML** page, add the following input elements to configure **Checkbox** widget.

### HTML

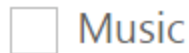
```
<div class="align">  
  <input type="checkbox" class="nodetext" id="checkbox_text" />  
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module CheckBoxComponent {  
  $(function () {  
    var checkbox = new ej.CheckBox($("#checkbox_text"), {  
      text: "Music"  
    });  
  });  
}
```

Execute the above code to render the following output.





### Rounded corner

Specifies the corner of **Checkbox** in rounded shape. Checkbox doesn't have rounded corner by default. To set rounded corner, you can enable **showRoundedCorner** property.

The following steps explains you the details about rendering the **Checkbox** with rounded corner.

In the **HTML** page, add the following input elements to configure **Checkbox** widget.

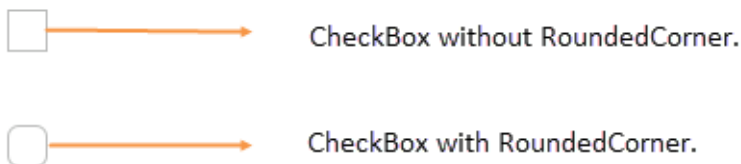
#### HTML

```
<div class="align">
<input type="checkbox" class="nodetext" id="checkbox_normalCorner" />
<br />
<br />
<input type="checkbox" class="nodetext" id="checkbox_roundedCorner" />
</div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CheckBoxComponent {
$(function () {
var checkbox = new ej.CheckBox($("#checkbox_normalCorner"), {
showRoundedCorner: false
});
var checkbox1 = new ej.CheckBox($("#checkbox_roundedCorner"), {
showRoundedCorner: true
});
});
}
```

Execute the above code to render the following output.



## Miscellaneous

### Checkbox Id

**Checkbox** id is not displayed in user interface. Here, **id** mentions the id attribute of the root element of **Checkbox** control. When you assign a value for **id** property, then this older id is replaced by new id. This id value should be unique.

Set id for **Checkbox** control as follows.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CheckBoxComponent {
  $(function () {
    var checkbox = new ej.CheckBox($("#check1"), {
      id: "sync"
    });
  });
}
```

### Checkbox Id Prefix

Id prefix value is the one that is appended to id value. It is used to mention the prefix for the wrapper's id attribute. When you assign a value for **idPrefix** property, the older prefix id is replaced by new prefix id.

Set prefix id for **Checkbox** control as follows.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CheckBoxComponent {
  $(function () {
    var checkbox = new ej.CheckBox($("#check1"), {
      idPrefix: "JS"
    });
  });
}
```

### Checkbox Name

The name attribute is used to identify from data after it is submitted to the server, or to reference from data using **JavaScript** on the client side. **Checkbox** also contains name attribute. This name should be a unique one. It is used to receive the **Checkbox** value. Without using name, you can't get the particular checkbox values at the time of submitting form.

### Checkbox Value

For **Checkboxes**, the contents of the value property do not appear in the user interface. The value property contains only a meaning when submitting a form. When a Checkbox is in checked state and when the form is submitted, the name of the **Checkbox** is sent along with the value of the value property (When the checkbox is not checked, no information is sent).

You can set name and value for **Checkbox** control as follows.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CheckBoxComponent {
$(function () {
var checkbox = new ej.CheckBox($("#check1"), {
name: "Conformation",
value : "received"
});
});
}
```

## CircularGauge

### Overview

The **CircularGauge** control visualizes the numerical values of scales in a circular manner. It is also a feature-rich control that provides extensive appearance customization options with support for the animation of a pointer element. The **CircularGauge** control comprises the following basic elements:

- Scales
- Pointers
- Ticks
- Labels
- Ranges
- Indicators

The gauge display can also be customized either as a full circle or a half circle based on individual requirements. Among other helpful features, the **Circular Gauge** control also includes advanced user interactivity.

### Key Features

- **Sub-Gauge:** Supports for rendering of multiple sub-gauges within the main Circular Gauge.
- **User Interaction:** Allows you to directly interact with the pointers of the gauge.
- **Indicators:** Supports for the indicator feature that shows the active or inactive state of the gauge.
- **Ranges:** Supports for highlighting the range of values in the gauge scale.
- **Pointers:** Supports for adding multiple pointers to the gauge.
- **Frametypes:** Supports two types of gauge displays, full circle and half circle.
- **Animation:** Supports the animation of a pointer.
- **CustomLabel:** Supports the addition of custom label text in the required location of the gauge.

### Getting Started

- This section encompasses how to configure Circular Gauge. You can provide data to a Circular Gauge and make them to display in a required way.
- The following screen shot displays a Circular Gauge, which visually represents the functions of an Automobile speedometer with RPM (Rotation per Minute), kph (Kilometer per hour) and it denotes the speed level indicators (Safe, Caution and Danger).



## Analog Speedometer

### Create a Circular Gauge

You can easily create the Circular Gauge widget by using the following steps.

1. First create a TypeScript project and the references in the app.ts page

For common getting started of TypeScript, you can refer [here](#).

The default type definition file **ej.web.all.d.ts** needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the jquery.d.ts file in your TypeScript application, which can be downloaded from [here](#).

2. Add the following scripts in the HTML page

### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
```

```
</html>
```

In the above code, `ej.web.all.min.js` script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

3. Create a

tag.

#### HTML

```
<html> <body> <div id="CircularGauge"></div> </body> </html>
```

4. Initialize the CircularGauge in ts file by using the `ej.CircularGauge` method.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CircularGaugeComponent {
  $(function () {
    var circularGaugeSample = new
    ej.datavisualization.CircularGauge($("#CircularGauge"));
  });
}
```

Run the above code example to get a default CircularGauge with default values.



#### Set Height and Width

Pointers have different height and width so you can set the height and width of the gauge according to your requirements. Set the basic values of the gauge such as height and width of the canvas element values that are to be rendered.

#### JAVASCRIPT

```
$(function () {
```

```
var circularGaugeSample = new  
ej.datavisualization.CircularGauge($("#CircularGauge"), {  
width: 500,  
height: 500  
});  
});
```

Run the above code example and you will see the following output.



### Set Background Color

The speedometer must have some dark color as background so that its value is clearly visible and you can vary the speed of the pointer by setting `ReadOnly` as false for user interaction.

### JAVASCRIPT

```
$(function () {  
var circularGaugeSample = new  
ej.datavisualization.CircularGauge($("#CircularGauge"), {  
width: 500,  
height: 500,  
backgroundColor: "#3D3F3D",  
readOnly: false,  
});  
});
```

Run the above code example and you will see the following output.



### Provide Scale Values

- The pointer cap can be customized with the following options. Cap radius, cap border color, cap background color, pointer cap border width are some of the properties that are customizable.
- The speed limit in the gauge has maximum value of 200 kph. So you can set maximum value for the gauge as 200.
- Major Ticks have the interval value of 20 and minor ticks have the interval value of 5. Show ranges and show indicators are used to display the ranges and indicators in their respective positions.

### JAVASCRIPT

```
$(function () {  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge"), {  
        width: 500,  
        height: 500,  
        backgroundColor: "#3D3F3D",  
        readOnly: false,  
        scales: [{  
            showRanges: true,  
            showIndicators: true,  
            pointerCap: {  
                radius: 15,  
                borderWidth: 0,  
                backgroundColor: "#797C79",  
                borderColor: "#797C79"  
            },  
            maximum: 200,  
            majorIntervalValue: 20,  
            minorIntervalValue: 5  
        }]  
    });  
});
```

Run the above code example and you will see the following output.



### Add Label Customization

To display the value around the scales, labels are used. By customizing the label color it displays as specified.

#### **JAVASCRIPT**

```
$(function () {  
  var circularGaugeSample = new  
  ej.datavisualization.CircularGauge($("#CircularGauge"), {  
    width: 500,  
    height: 500,  
    backgroundColor: "#3D3F3D",  
    readOnly: false,  
    scales: [{  
      //Add the labels customization code here  
      labels: [{  
        color: "#ffffff"  
      }],  
    }];  
  });  
});
```

Run the above code example and you will see the following output.





### Add Pointers

Here, you have three pointers that denote the kilometer value, rotation per minute value and torque value. The torque value pointer needs not be similar to the other two pointers. You can set torque pointer as marker pointer. And you can set other attributes for pointer such as background color, border color, length, width and distance from scale.

### JAVASCRIPT

```
$(function () {  
  var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge"), {  
    width: 500,  
    height: 500,  
    backgroundColor: "#3D3F3D",  
    readOnly: false,  
    scales: [{  
      //Add the labels customization code here  
      //Add the pointers customization code here  
    }, {  
      pointers: [{  
        value: 140,  
        distanceFromScale: 60,  
        showBackNeedle: false,  
        length: 20,  
        type: "marker",  
        markerType: "triangle",  
        width: 10,  
        radius: 10,  
        backgroundColor: "#FF940A",  
        border: {  
          color: "#FF940A"  
        },  
      },  
    ],  
    {  
      value: 110,  
      showBackNeedle: false,  
      length: 150,  
    }  
  }  
});
```

```
width: 2,  
radius: 10,  
needleType: "rectangle",  
backgroundColor: "#05AFFE",  
border: {  
  color: "#05AFFE"  
},  
, {  
  value: 67,  
  showBackNeedle: false,  
  length: 100,  
  width: 15,  
  radius: 10,  
  backgroundColor: "#FC5D07",  
  border: {  
    color: "#FC5D07"  
  },  
},  
}],  
//Add the ticks customization code here  
//Add the ranges customization code here  
//Add the indicators customization code here  
//Add the Custom labels customization code here  
}]  
});  
});
```

Run the above code example and you will see the following output.



### Add Tick Details

- You can set Major ticks with their width and height equal to Minor ticks.
- You can set Color according to your preference for better visibility in dark backgrounds.
- To display and customize the tick value add the following code example.

### JAVASCRIPT

```
$(function () {  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge"), {  
        width: 500,  
        height: 500,  
        backgroundColor: "#3D3F3D",  
        readOnly: false,  
        scales: [{  
            //Add the labels customization code here  
            //Add the pointers customization code here  
            //Add the ticks customization code here  
            ticks: [{  
                type: "major",  
                distanceFromScale: 70,  
                height: 20,  
                width: 3,  
                color: "#ffffff"  
            }, {  
                type: "minor",  
                height: 12,  
                width: 1,  
                distanceFromScale: 70,  
                color: "#ffffff"  
            }]  
        }];  
    });  
});
```

Run the above code example and you will see the following output.



### Add Range Values

- Ranges denote the property of the scale value in the speedometer. The color values of the ranges denote speed variation. Set ShowRanges as true for showing the ranges in the Circular Gauge.

- For Low speed, you can mention it as safe zone; for moderate speed, you can call it as caution zone and for high speed, you can mark it as high speed.
- You can customize the range with properties such as start value, end value, start width, end width, background color , border color, etc.,

### JAVASCRIPT

```
$(function () {  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge"), {  
        width: 500,  
        height: 500,  
        backgroundColor: "#3D3F3D",  
        readOnly: false,  
        scales: [{  
            ranges: [{  
                distanceFromScale: 30,  
                startValue: 0,  
                endValue: 70,  
                backgroundColor: "#5DF243",  
                border: {  
                    color: "#FFFFFF"  
                },  
            }, {  
                distanceFromScale: 30,  
                startValue: 70,  
                endValue: 140,  
                backgroundColor: "#F6FF0A",  
                border: {  
                    color: "#FFFFFF"  
                },  
            },  
            {  
                distanceFromScale: 30,  
                startValue: 140,  
                endValue: 200,  
                backgroundColor: "#FF1807",  
                border: {  
                    color: "#FFFFFF"  
                },  
            },  
        ]],  
    }]);  
});
```

Run the above code example and you will see the following output.



### Add Indicator Details

- Indicators denote whether the pointers values are in their respective zones or not. Positioning the indicator on the respective range value gives you the required changes.
- By using Position property, you can set the location of the indicator. StateRanges defines how the indicator should behave when the pointer is in certain values.

### JAVASCRIPT

```
$(function () {  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge"), {  
        width: 500,  
        height: 500,  
        backgroundColor: "#3D3F3D",  
        readOnly: false,  
        scales: [{  
            indicators: [  
                {  
                    height: 10,  
                    width: 10,  
                    type: "circle",  
                    position: { x: 210, y: 300 },  
                    stateRanges: [{  
                        endValue: 70,  
                        startValue: 0,  
                        backgroundColor: "#5DF243",  
                        borderColor: "#5DF243",  
                        text: "",  
                        textColor: "#870505"  
                    }, {  
                        endValue: 200,  
                        startValue: 70,  
                        backgroundColor: "#145608",  
                        borderColor: "#145608",
```

```
text: "",
textColor: "#870505"
}],
{
height: 10,
width: 10,
type: "circle",
position: { x: 255, y: 200 },
stateRanges: [{
endValue: 140,
startValue: 70,
backgroundColor: "#F6FF0A",
borderColor: "#F6FF0A",
text: "",
}, {
endValue: 70,
startValue: 0,
backgroundColor: "#969B0C",
borderColor: "#969B0C",
text: "",
}, {
endValue: 200,
startValue: 140,
backgroundColor: "#969B0C",
borderColor: "#969B0C",
text: "",
}]
}, {
height: 10,
width: 10,
type: "circle",
position: { x: 300, y: 300 },
stateRanges: [{
endValue: 140,
startValue: 0,
backgroundColor: "#890F06",
borderColor: "#890F06",
text: "",
},
{
endValue: 200,
startValue: 140,
backgroundColor: "#FF1807",
borderColor: "#FF1807",
text: "",
}]
}],
});
});
```

Run the above code example and you will see the following output.



### Add Custom Label Details

Custom labels are used to specify the texts that need to be displayed in the gauge. You can customize it through various properties. To display the three range description, custom texts are used here.

### JAVASCRIPT

```
$(function () {
    var circularGaugeSample = new
    ej.datavisualization.CircularGauge($("#CircularGauge"), {
        width: 500,
        height: 500,
        backgroundColor: "#3D3F3D",
        readOnly: false,
        scales: [{
            customLabels: [{
                value: "Safe",
                position: { x: 200, y: 280 },
                color: "#5DF243",
                font:
                {
                    size: "12px",
                    fontFamily: "Arial",
                    fontStyle: "Bold"
                }
            }, {
                value: "Caution",
                position: { x: 253, y: 212 },
                color: "#F6FF0A",
                font:
                {
                    size: "12px",
                    fontFamily: "Arial",
                    fontStyle: "Bold"
                }
            }, {
                value: "Danger",
                position: { x: 290, y: 280 },
```

```

color: "#FF1807",
font:
{
size: "12px",
fontFamily: "Arial",
fontStyle: "Bold"
}
}]
}]
});
});

```

Run the above code example and you will see the following output.



## Interaction and Animation

### Interaction

**Circular Gauge** control contains **Interaction** feature. You can use this interaction feature to change the pointer values manually. You can achieve this by clicking and dragging the pointer over the **Gauge** and you can see the value of pointer changes dynamically while dragging. To Enable/Disable the user interaction you can use the Boolean property called `readOnly`. The user interaction option is enabled when you set the value as false for the property `readOnly`. By default it holds the true value. That is by default it does not support interaction.

### HTML

```
<div id="CircularGauge1"> </div>
```

### JAVASCRIPT

```

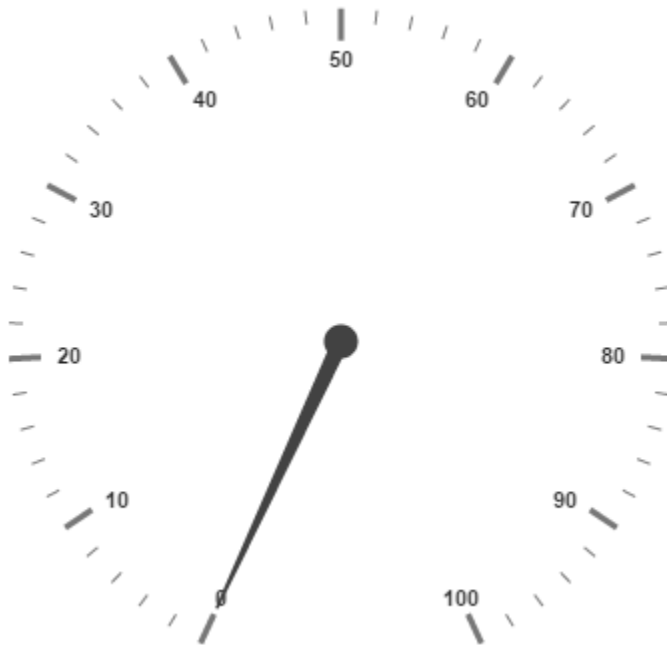
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CircularGaugeComponent {
$(function () {
// For Circular Gauge rendering

```



```
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
// For User interaction
readOnly: false,
})
});
```

Execute the above code to render the following output.



### Animations

**Circular Gauge** contains an attractive concept called **Animation**. The animation option enables the pointer to rotate from the minimum value to the current value with animation effects. By using this animation you can change the pointer value dynamically. You can apply the animation on pointer either by clockwise or counterclockwise based on the scale direction. You can enable / disable it using the property `enableAnimation`. Animation is enabled when you set `enableAnimation` as 'true'. By default it holds the true value. You can control the speed of the pointer during animating by using the property `animationSpeed`. It is a numerical value that holds the time in milliseconds. That is when the value is given as 1000, it is considered as 1 second.

### HTML

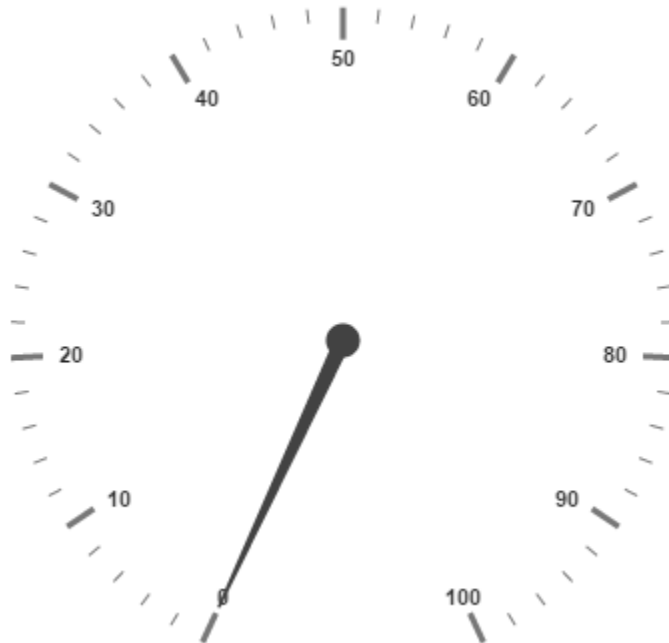
```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {
// For Circular Gauge rendering
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
// For enabling animation
```

```
enableAnimation: true,  
// For setting animation speed  
animationSpeed:1000  
})  
});
```

Execute the above code to render the following output.



### Gradient

You can change the interior gradient of circular gauge by using `interiorGradient` property. The `isRadialGradient` property is used to check whether the gradient is circular or not.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
  var circularGaugeSample = new  
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
    interiorGradient: { colorInfo: [{ colorStop : 0, color: "#FFFFFF" }, { colorStop :  
    1, color: "#AAAAAA" } ] },  
    isRadialGradient : true,  
  })  
});
```

### Distance From Corner

You can display the circular gauge from distance apart from the corner by specifying value for `distanceFromCorner` property.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
        distanceFromCorner : 25  
    })  
});
```

### Resize

Circular gauge can be responsive while resizing by specifying `enableResize` property as true.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
        enableResize: true  
    })  
});
```

### Localization

The circular gauge can be localized based on name of culture specified in `locale` property.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
        locale : "en-US"  
    })  
});
```

### Themes

**CircularGauge** theme is a set of pre-defined options that are applied to the control before **CircularGauge** is instantiated. Following predefined themes are available in JavaScript **CircularGauge**.

1. flat light
2. flat dark

3. gradient light
4. gradient dark
5. azure
6. azure dark
7. lime
8. lime dark
9. saffron
10. saffron dark
11. gradient azure
12. gradient azure dark
13. gradient lime
14. gradient lime dark
15. gradient saffron
16. gradient saffron dark

The theme for circular gauge can be specified using `theme` property.

#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {  
  var circularGaugeSample = new  
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
    theme : "flatlight"  
  })  
});
```

#### Circular Gauge Values

The `minimum`, `maximum`, `radius` and `value` attributes of circular gauge are used to render the circular gauge with specified location.

#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {  
  var circularGaugeSample = new  
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
    maximum: 120,  
    minimum: 10,  
    value: 30,  
    radius: 100  
  })  
});
```

## Scales

**Scales** are the basic functional block of the **Circular Gauge**. By customizing the **scales**, the appearance of the **Gauge** can be improved. The functional blocks of Circular Gauge are

- Pointers
- Labels
- CustomLabels
- Indicators
- Ticks
- Ranges
- Sub gauges.

### Adding Scale Collection

**Scale collection** is directly added to the **Gauge** object. Refer the following code example to add scale collection in **Gauge** control. You customize the scale radius using **radius** property.

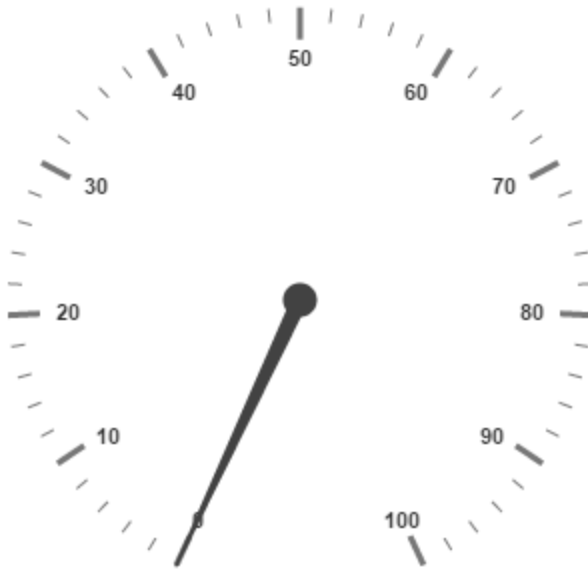
#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CircularGaugeComponent {
    $(function () {
        //For circular gauge rendering
        var circularGaugeSample = new
        ej.datavisualization.CircularGauge($("#CircularGauge1"), {
            scales: [{
                radius: 150
            }]
        })
    });
}
```

Execute the above code to render the following output.



### Scale Customization

#### Colors and Border

The Scale border is modified with the object called **border**. It has two border property namely **color** and **width** which are used to customize the border color of the scale and border width of the scale. Setting the background color improves the look and feel of the **Circular Gauge**. You can customize the background color of the scale using **backgroundColor**. The scale bar of circular gauge can be enabled by setting **showScaleBar** property as true.

#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {  
    //For circular gauge rendering  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
        scales: [{  
            showScaleBar: true,  
            radius: 150,  
            backgroundColor: "Red",  
            border: {  
                //For scale border color  
                color: "Blue",  
                //For scale border width  
                width: 3  
            },  
            pointers: [{ length: 100 }]  
        }]  
    })  
});
```

Execute the above code to render the following output.



### Pointer Cap

**Pointer cap** is a circular shape element that is located at the center of the **Circular Gauge**. The pointer cap is one of the cynosure of the **Circular Gauge**. By customizing the pointer cap, Gauge style is improved. The pointer cap is modified with the object `pointerCap`. It contains `radius`, `borderColor`, `borderWidth`, `interiorGradient` and `backgroundColor` properties. The property `radius` is used to set the radius for the pointer cap.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {
    // For Circular Gauge rendering
    var circularGaugeSample = new
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {
        scales: [{ pointerCap: {
            // For setting pointer cap radius
            radius:10,
            // For setting pointer cap border width
            borderWidth: 4,
            // For setting pointer cap border color
            borderColor: "Blue",
            // For setting pointer cap background color
            backgroundColor: "Red"
        }}]
    })
});
```

Execute the above code to render the following output.



### Appearance

**Circular Gauge** contains two types of scale direction such as clockwise and counter clockwise. You can set them by enumerable property called `direction`. And you can set the minimum and maximum values for the scale with the properties `minimum` and `maximum*`. The two properties `minorIntervalValue` and `majorIntervalValue` are the values used to set interval value for the ticks and labels. The `radius` property is used to set the radius value for the circular scale and the `size` property is used to set the scale bar width. You can also adjust the Opacity of the scale with the property `opacity`. The value for opacity lies between 0 and 1. You can also give some shadow effects for the scale by using the property `shadowOffset`.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
  // For Circular Gauge rendering  
  var circularGaugeSample = new  
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
    scales: [{  
      // For setting scale bar size  
      size: 30,  
      // For setting scale radius  
      scaleRadius: 130,  
      // For setting scale minimum value  
      minimum: 20,  
      // For setting scale maximum value  
      maximum: 120,  
      // For setting scale major interval value  
      majorIntervalValue: 20,
```



```
// For setting scale minor interval
minorIntervalValue: 5,
// For setting scale direction
direction:ej.datavisualization.CircularGauge.Directions.CounterClockwise,
// For setting scale background color
backgroundColor:"red",
// For setting scale bar opacity
opacity:0.5,
// For setting scale bar shadow offset
shadowOffset: 20
}]
});
});
```

Execute the above code to render the following output.



### Circular Gauge Scale Angles

The property `startAngle` is used to set starting position of the scale at certain angle and `sweepAngle` is used to shrink or expand the scale to certain angle.

#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {
// For Circular Gauge rendering
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
scales: [{
startAngle: 90,
sweepAngle: 200
}]
});
```

```
});  
});
```

### Enable/Disable properties

You can enable / disable properties in Circular Gauge using some properties in scale collection. The `showIndicators` property is used to enable/disable the indicators. `showLabels`, `showTicks`, `showRanges`, `showPointers` and `showScaleBar` are used to enable/ disable labels, ticks, ranges, pointers and scale bar respectively.

### Multiple Scales

You can set **Multiple scales** for a single **Circular Gauge** control by using an array of scale objects. Each scale object is independent of each other. The following code example refers to two scale objects in a Gauge.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    // For Circular Gauge rendering  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
        scales: [  
            // For setting scale1  
            {  
                size: 10,  
                showScaleBar: true,  
                scaleRadius: 150,  
                minimum: 20,  
                maximum: 120,  
                majorIntervalValue: 20,  
                minorIntervalValue: 5,  
                direction: ej.datavisualization.CircularGauge.Directions.Clockwise,  
                shadowOffset: 20,  
                pointers: [{value: 50, length: 120}]  
            },  
            // For setting second scale  
            {  
                size: 10,  
                showScaleBar: false,  
                scaleRadius: 80,  
                majorIntervalValue: 10,  
                direction: ej.datavisualization.CircularGauge.Directions.CounterClockwise,  
                opacity: 0.5,  
                shadowOffset: 5,  
                pointers: [{value: 40, length: 50}],  
                ticks: [{color: "red", distanceFromScale: 80}],  
                labels: [{ distanceFromScale: 40, color: "red"}]  
            }  
        ]  
    });  
});
```

Execute the above code to render the following output.



### Pointers

**Pointer** value points out the actual value set in the **Circular Gauge**. You can customize the **pointers** to improve the appearance of **Gauge**.

### Adding Pointer Collection

**Pointer collection** is directly added to the scale object. To add pointer collection in a **Gauge** control refer the following code example.

#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {  
    //For circular gauge rendering  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
        scales: [{  
            pointers: [{  
                value: 30  
            }]  
        }]  
    });  
});
```

Execute the above code to render the following output.



### Adding Pointer Value

**Pointer value** is the important element in the **Circular Gauge** that indicates the **Gauge** value. Real purpose of the **Circular Gauge** is based on the pointer value. You can set the pointer value either directly during rendering the control or it can be achieved by public method too.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    // For Circular Gauge rendering  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
        scales: [{  
            showRanges: true,  
            showScaleBar: true,  
            radius: 150, size: 2,  
            ranges: [{  
                startValue: 20,  
                endValue: 80,  
                backgroundColor: "Green",  
            }],  
            pointers: [{  
                value: 30  
            }]  
        }]  
    });  
});
```

Execute the above code to render the following output.



### Pointer Styles

#### Colors and Border

The Pointers border is modified with the object called `border` as in scales. It has two border property called `color` and `width` which are used to customize the border color of the pointer and border width of the pointer. You can set the background color to improve the look of the **Circular Gauge** and you can customize the background color of the scale using `backgroundColor`. The needle length of circular gauge can be customized using `backNeedleLength` property.

#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CircularGaugeComponent {
$(function () {
// For Circular Gauge rendering
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
scales: [{
showScaleBar: true,
width: 10, radius: 150,
pointers: [{
// For setting pointer border
border: { color: "green", width: 2 },
// For setting pointer background
backgroundColor: "yellow",
// For setting pointer back needle length
backNeedleLength: 10,
// For setting pointer value
value: 45,
// For setting pointer length
```

```
length: 100,  
// For setting pointer width  
width: 16,  
// For setting pointer opacity  
opacity: 0.6  
}]  
}]  
});  
});  
}
```

Execute the above code to render the following output.



### Appearance

Based on the value, the **pointer** point out the label value. You can set the pointer length and width using **length** and **width** property respectively. And you can also adjust the opacity of the pointer using the property **opacity** which holds the value between 0 and 1. You can add the gradient effects to the pointer using **gradients** object.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
  // For Circular Gauge rendering  
  var circularGaugeSample = new  
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
    scales: [{  
      showScaleBar: true,  
      backgroundColor: "orange",  
      border: { width: 2, color: "Red" },  
      width: 10, radius: 150,
```

```

pointers: [{
  // For setting pointer border
  border: { color: "red", width: 2 },
  // For setting pointer background
  backgroundColor: "orange",
  // For setting pointer value
  value: 45,
  // For setting pointer length
  length: 100,
  // For setting pointer width
  width: 16,
  // For setting pointer opacity
  opacity: 0.6
}]
});

```

Execute the above code to render the following output.



### Position the pointer

Pointer can be positioned with the help of two properties such as **distanceFromScale** and **placement**. **distanceFromScale** property defines the distance between the scale and pointer. **placement** property is used to locate the pointer with respect to scale either inside the scale or outside the scale or along the scale. It is an enumerable data type. Both the property is applied only if pointer **type** is marker. For **needleType** marker, it renders with default position that is unchangeable.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function() {
```

```
// For Circular Gauge rendering
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
  scales: [{
    showScaleBar: true,
    backgroundColor: "orange",
    border: {
      width: 2,
      color: "red"
    },
    width: 10,
    radius: 150,
    pointers: [{
      type: "marker",
      // For setting marker type
      markerType: "triangle",
      // For setting pointer position
      placement: "near",
      // For setting pointer distance from scale
      distanceFromScale: 10,
      // For setting pointer border
      border: {
        color: "red",
        width: 2
      },
      // For setting pointer background
      backgroundColor: "orange",
      // For setting pointer value
      value: 40,
      // For setting pointer length
      length: 20,
      // For setting pointer width
      width: 20,
      // For setting pointer opacity
      opacity: 0.6
    }]
  }]
});
```

Execute the above code to render the following output.





### Types

- Circular gauge pointer has two types such as,
  - Needle
  - Marker
- Needle type pointers are the default pointers that cannot be positioned and that is located at the center of the gauge. There are four different shapes of needle pointers such as
  - Rectangle
  - Triangle
  - Trapezoid
  - Arrow
  - Image
- For marker pointer, the available dimensions are
  - Rectangle
  - Triangle
  - Ellipse
  - Diamond
  - Pentagon
  - Circle
  - Slider
  - Pointer
  - Wedge
  - Trapezoid
  - Rounded Rectangle
  - Image

### Pointer Images

In JavaScript circular gauge, it is possible to replace the pointer with images. We can use image instead of rendering the pointer.

### Image with URL

- To implement the pointer image we need to give the API called `imageUrl`. It is a string data type.
- Image type pointer is applicable for both marker as well as needle type pointers and it is possible with combine the normal marker pointer type with image type. The three possibilities are
  - Needle Image
  - Marker Image
  - Marker pointer with Image

### Needle Image

In `needleType`, needle pointer is completely replaced by image. We can implement it with the help of following snippet.

#### HTML

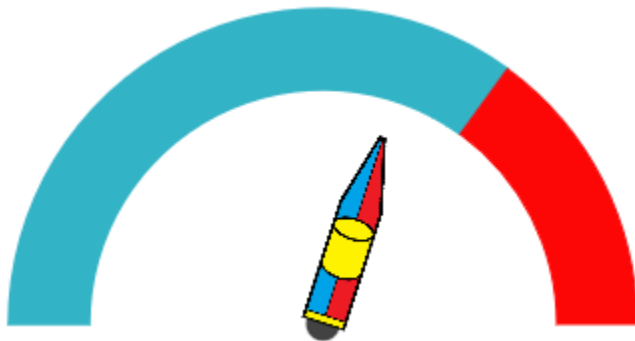
```
<div id="CoreCircularGauge"></div>
```

#### JAVASCRIPT

```
$(function() {  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
        // To set frame type as half circle  
        frame: {  
            frameType: "halfcircle"  
        },  
        // To set scale options  
        scales: [{  
            // set basic appearance  
            showRanges: true,  
            showLabels: false,  
            startAngle: 180,  
            sweepAngle: 180,  
            radius: 130,  
            showScaleBar: false,  
            // To set pointer option  
            pointers: [{  
                // To set pointer type as needle  
                type: "needle",  
                // To set needle type as image  
                needleType: "image",  
                // To set image url for pointer image  
                imageUrl: "nib.png",  
                // To set pointer value  
                value: 60,  
                // To set pointer dimension  
                length: 30,  
                width: 100,  
            }],  
            // To set tick options  
            ticks: [{  
                height: 0,  
                width: 0  
            }],  
        }  
    });  
});
```

```
// To set range options
ranges: [{
  distanceFromScale: -30,
  startValue: 0,
  endValue: 70,
  size: 40,
}, {
  distanceFromScale: -30,
  startValue: 70,
  endValue: 110,
  backgroundColor: "#fc0606",
  border: {
    color: "#fc0606"
  },
  size: 40,
}]
});
```

Execute the above code to render the following output.



### Marker Image

In `markerType`, marker pointer is completely replaced by image. We can implement it with the help of following snippet.

### HTML

```
<div id="CoreCircularGauge"></div>
```

### JAVASCRIPT

```
$(function () {
  var circularGaugeSample = new
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {
    // To set frame type as half circle
    frame: {
      frameType: "halfcircle"
    },
    // To set scale options
    scales: [{
      // set basic appearance
      showRanges: true,
```

```
showLabels: false,
startAngle: 180,
sweepAngle: 180,
radius: 130,
showScaleBar: false,
// To set pointer option
pointers: [{
  // To set pointer type as marker
  type: "marker",
  // To set needle type as image
  markerType: "image",
  // To set image url for pointer image
  imageUrl: "ball.png",
  // To set pointer value
  value: 60,
  // To set pointer dimension
  length: 30,
  width: 100,
}],
// To set tick options
ticks: [{
  height: 0,
  width: 0
}],
// To set range options
ranges: [{
  distanceFromScale: -30,
  startValue: 0,
  endValue: 70,
  size: 40,
}, {
  distanceFromScale: -30,
  startValue: 70,
  endValue: 110,
  backgroundColor: "#fc0606",
  border: {
    color: "#fc0606"
  },
  size: 40,
}]
});
});
```

Execute the above code to render the following output.



### Marker Pointer with Image

In this type, marker pointer is drawn first and then image also loaded. We can implement it with the help of following snippet.

#### HTML

```
<div id="CoreCircularGauge"></div>
```

#### TS

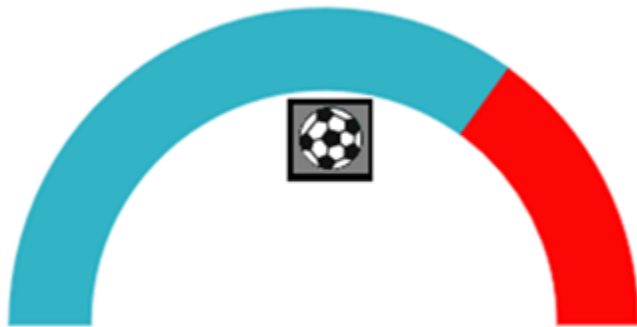
```
$ (function () {
  var circularGaugeSample = new
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {
    // To set frame type as half circle
    frame: { frameType: "halfcircle" },
    // To set scale options
    scales: [{
      // set basic appearance
      showRanges: true, showLabels: false,
      startAngle: 180, sweepAngle: 180, radius: 130,
      showScaleBar: false,
      // To set pointer option
      pointers: [{
        // To set pointer type as marker
        type: "marker",
        // To set needle type as rectangle
        markerType: "rectangle",
        // To set image url for pointer image
        imageUrl: "ball.png",
        // To set pointer value
        value: 50,
        // To set pointer dimension
        length: 30,
        width: 100,
        border: { color: "Black", width: 3 }
      }],
      // To set tick options
      ticks: [{ height: 0, width: 0 }],
      // To set range options
      ranges: [{
        distanceFromScale: -30,
        startValue: 0,
        endValue: 70, size: 40,
      }, {
```

```

distanceFromScale: -30,
startValue: 70,
endValue: 110,
backgroundColor: "#fc0606",
border: { color: "#fc0606" }, size: 40,
}]
}]
});
});

```

Execute the above code to render the following output.



### Multiple Pointers

**Circular Gauge** can have multiple pointers on it. You can use any combination and any number of pointers in a **Gauge**. That is, a Gauge can contain any number of marker pointer and any number of needle pointers. Refer the following code example containing two pointers.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```

$(function () {
// For Circular Gauge rendering
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
scales: [{
showScaleBar: true,
backgroundColor: "#DCEBF9",
border: { width: 2, color: "Green" },
width: 10, radius: 150,
pointers: [
// For setting pointer1
{
border: { color: "Green", width: 2 },
backgroundColor: "#DCEBF9",
value: 40,
length: 100,
width: 16,
opacity: 0.6
},
// For setting pointer2
{

```

```

distanceFromScale: 20,
placement: "near",
type: "marker",
markerType: "triangle",
length: 20,
width: 20,
value: 60,
backgroundColor: "#DCEBF9",
border: { color: "Green", width: 2 },
}]
}]
});
});

```

Execute the above code to render the following output.



#### Pointer Value Text

Gauge `pointerValueText` is used to display the current value of the pointer in the **Circular Gauge** control.

#### Positioning the text

You can position the **Circular Gauge** pointer value with the gauge as center by using the **API** called `distance`. You can Disable/ Enable these pointers value by using the API `showValue`.

#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```

$(function () {
    var circularGaugeSample = new
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {
        // Setting basic properties
    }

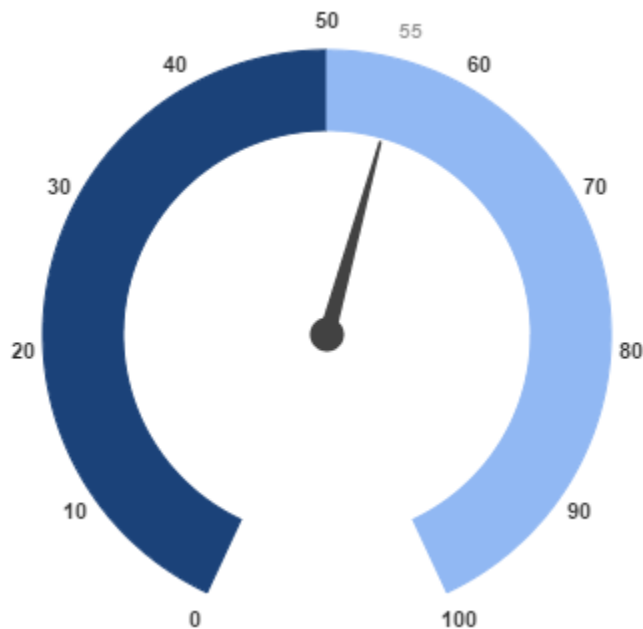
```

```

radius: 100, value: 55, backgroundColor: "transparent",
// Setting scale values
scales: [{
  showRanges: true,
  // Setting tick properties
  ticks: [{ height: 0, width: 0 }],
  // Setting range properties
  ranges: [
    { size: 40, startValue: 0, endValue: 50, backgroundColor: "#1B4279", border:
      { color: "#1B4279" } },
    { size: 40, startValue: 50, endValue: 100, backgroundColor: "#91B8F3",
      border: { color: "#91B8F3" } }
  ],
  // Setting pointer properties
  pointers: [{
    // Setting pointer value text properties
    pointerValueText: {
      // enable showValue property
      showValue: true,
      // setting distance property
      distance: 0,
      color: "#8c8c8c" }
    },
    },
  ]
});

```

Run the above code to render the output as follows.



### Appearance

Appearance of the **Circular Gauge** `pointerValueText` is adjusted by using four properties. Such as **color**, **angle**, **autoAngle** and **opacity**.



- `color` property is used to set the color of the pointer value text.
- `angle` property is used to set the angle in which the text is displayed.
- `autoAngle` is used to display the text in certain angle based on pointer position angle.
- `opacity` is used to customize the brightness of the text.

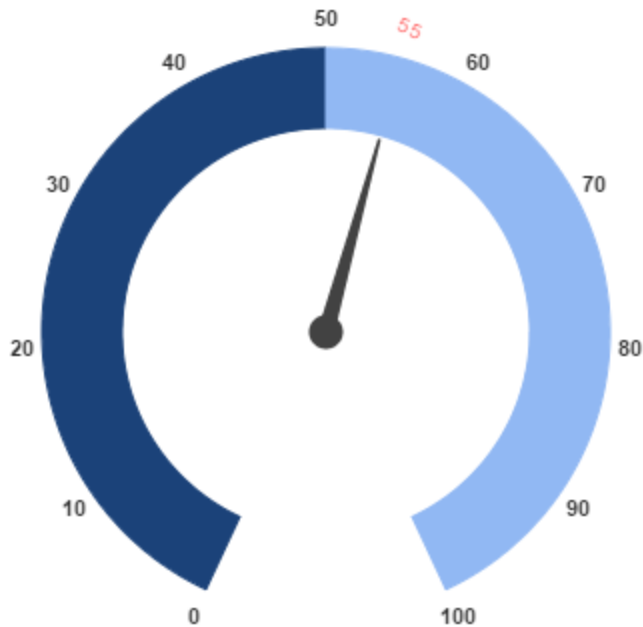
## HTML

```
<div id="CircularGauge1"></div>
```

## JAVASCRIPT

```
$(function () {  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
        // Setting basic properties  
        radius: 100, value: 55, backgroundColor: "transparent",  
        // Setting scale values  
        scales: [{  
            showRanges: true,  
            // Setting tick properties  
            ticks: [{ height: 0, width: 0 }],  
            // Setting range properties  
            ranges: [  
                { size: 40, startValue: 0, endValue: 50, backgroundColor: "#1B4279", border:  
                { color: "#1B4279" } },  
                { size: 40, startValue: 50, endValue: 100, backgroundColor: "#91B8F3",  
                border: { color: "#91B8F3" } }  
            ],  
            // Setting pointer properties  
            pointers: [{  
                // Setting pointer value text properties  
                pointerValueText: {  
                    showValue: true,  
                    distance: 0,  
                    // Setting color property  
                    color: "Red",  
                    // Setting opacity property  
                    opacity: 0.7,  
                    // Setting angle property  
                    angle: 20  
                }  
            }],  
        }],  
    });  
});
```

Run the above code to render the output as follows.



### Font Options

Similar to other collection, **font** option is also available in this pointer value text such as **size**, **fontFamily** and **fontStyle**.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

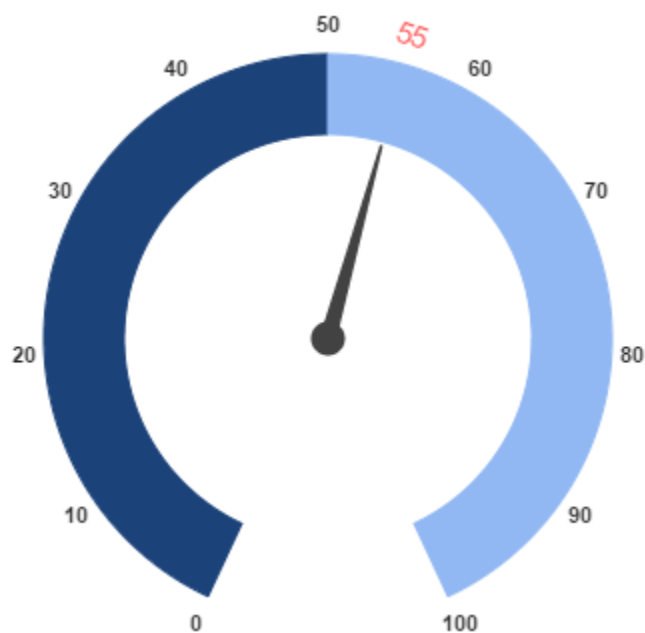
```
$(function () {
    var circularGaugeSample = new
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {
        // Setting basic properties
        radius: 100, value: 55, backgroundColor: "transparent",
        // Setting scale values
        scales: [{
            showRanges: true,
            // Setting tick properties
            ticks: [{ height: 0, width: 0 }],
            // Setting range properties
            ranges: [
                { size: 40, startValue: 0, endValue: 50, backgroundColor: "#1B4279", border:
                { color: "#1B4279" } },
                { size: 40, startValue: 50, endValue: 100, backgroundColor: "#91B8F3",
                border: { color: "#91B8F3" } }
            ],
            // Setting pointer properties
            pointers: [{
                // Setting pointer value text properties
                pointerValueText: {
                    showValue: true,
                    distance: 0,
                    color: "Red",
                    opacity: 0.7,
```

```

angle: 20,
//setting font option
font: {
  size: "15px",
  fontStyle: "Normal",
  fontFamily: "Arial",
}
},
},
});
});

```

Run the above code to render the output as follows.



## Labels

**Labels** are units that are used to display the values in the scales. You can customize **labels** with the properties like **angle**, **color**, **font**, **opacity**, etc.

## Adding Label Collection

**Label collection** is directly added to the scale object. Refer the following code example to add label collection in a **Gauge**.

## HTML

```
<div id="CircularGauge1"></div>
```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CircularGaugeComponent {
  $(function () {

```

```
//For circular gauge rendering
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
  scales: [{
    labels: [{
      angle: 30
    }]
  }]
});
```

Execute the above code to render the following output.



### Label Customization

#### Appearance

The **attribute** `angle` is used to display the labels in the specified angles and **color** attribute is used to display the labels in specified color. You can adjust the opacity of the label with the property `opacity` and the values of it lies between 0 and 1. You can adjust the labels based on the tick's direction by setting `autoAngle` as true. `includeFirstValue` is a special property especially used in some special scenarios such as in clock, where the value 0 needs to be replaced with that of 12. By enabling this property the first value of the label is not rendered.

Font option is also available on the labels. The basic three properties of **font** such as size, family and style can be achieved by `size`, `fontStyle` and `fontFamily`. Labels are two types such as major and minor. Major types labels are for major interval values and minor types labels are for minor interval values.

#### HTML

```
<div id="CircularGauge1"></div>
```

**JAVASCRIPT**

```
$(function () {
  // For Circular Gauge rendering
  var circularGaugeSample = new
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {
    scales: [{
      showScaleBar: true,
      backgroundColor: "#FAF4B5",
      border: { width: 2, color: "Yellow" },
      width: 10, radius: 150,
      pointers: [{
        border: { color: "Yellow", width: 2 },
        backgroundColor: "#FAF4B5",
        value: 40, length: 100,
        width: 16,
        opacity: 0.6
      }],
      labels: [{
        // For setting label angle
        angle: 10,
        // For setting label opacity
        opacity: 0.8,
        // For disable the include first value property
        includeFirstValue: false,
        // For setting label color
        color: "Yellow",
        // For setting label font
        font: {
          size: "15px",
          fontFamily: "Arial",
          fontStyle: "bold"
        }
      }],
    }],
  });
});
```

Execute the above code to render the following output.



### Unit text and Position

`unitText` is used to add some text along with the labels. For example, in speedometer, you need to mention the units in kph. You can also add the unit text in front of the labels. You can achieve this by using an enumerable property `unitTextPosition`. With this you can position the unit text in front or back.

Labels can be positioned with the help of two properties such as `distanceFromScale` and `placement`. **`distanceFromScale`** property defines the distance between the scale and labels. **`placement`** property is used to locate the labels with respect to scale either inside the scale or outside the scale or along the scale. It is an enumerable data type.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

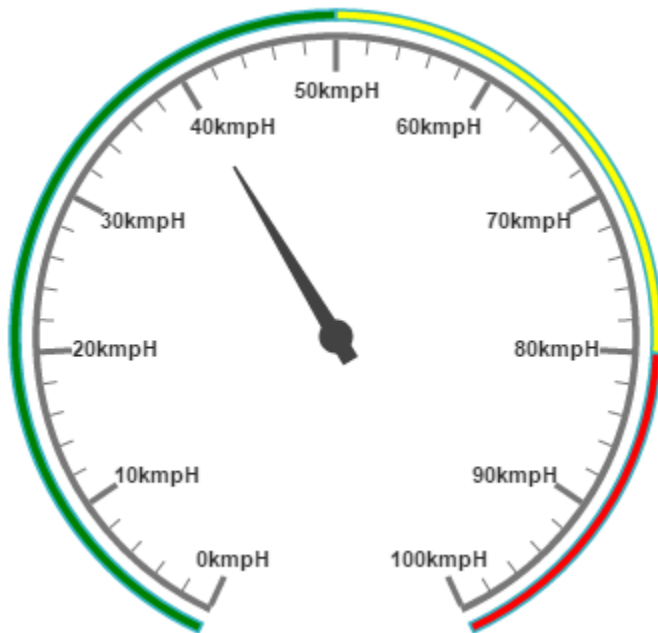
```
$(function () {  
    // For Circular Gauge rendering  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
        scales: [{  
            showRanges: true,  
            showScaleBar: true,  
            radius: 150, size: 2,  
            pointers: [{  
                value: 40,  
                showBackNeedle: true,  
                length: 100  
            }],  
            labels: [{  
                // For setting unit text  
                unitText: "kmpH",  
                // For setting unit text position
```

```

unitTextPosition: "back"
}],
ranges: [{ startValue: 0, endValue: 50, backgroundColor: "Green", placement:
"far", distanceFromScale: -30 },
{ startValue: 50, endValue: 80, backgroundColor: "yellow", placement: "far",
distanceFromScale: -30 },
{ startValue: 80, endValue: 100, backgroundColor: "red", placement: "far",
distanceFromScale: -30 }]
}]
});
});

```

Execute the above code to render the following output.



### Multiple Labels

You can achieve multiple labels such as minor and major **type** in a **Gauge** sample scale. Refer the following code example for multiple labels variation.

#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```

$(function () {
// For Circular Gauge rendering
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
scales: [{
showRanges: true, minorIntervalValue: 5,
backgroundColor: "yellow",
border: { width: 1.5, color: "Red" },
showScaleBar: true, radius: 150, size: 5,
pointerCap: {

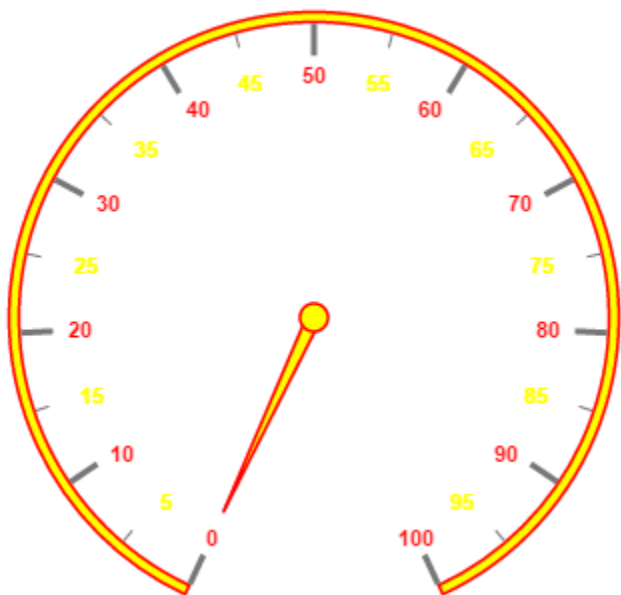
```

```

backgroundColor: "yellow",
borderColor: "Red", borderWidth: 1.5
},
labels: [
  // For setting label1
  { type: "minor", color: "yellow" },
  // For setting label2
  { type: "major", color: "Red" }],
pointers: [{
  backgroundColor: "yellow",
  border: { width: 1.5, color: "Red" },
  length: 110
}]
}]
});
});

```

Execute the above code to render the following output.



## Ticks

The ticks are used to mark some values on the scale. Based on the tick's value you can set the labels on the required position.

## Adding Tick Collection

Tick collection is directly added to the scale object. Refer the following code example to add tick collection in a **Gauge** control.

## HTML

```
<div id="CircularGauge1"></div>
```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />

```

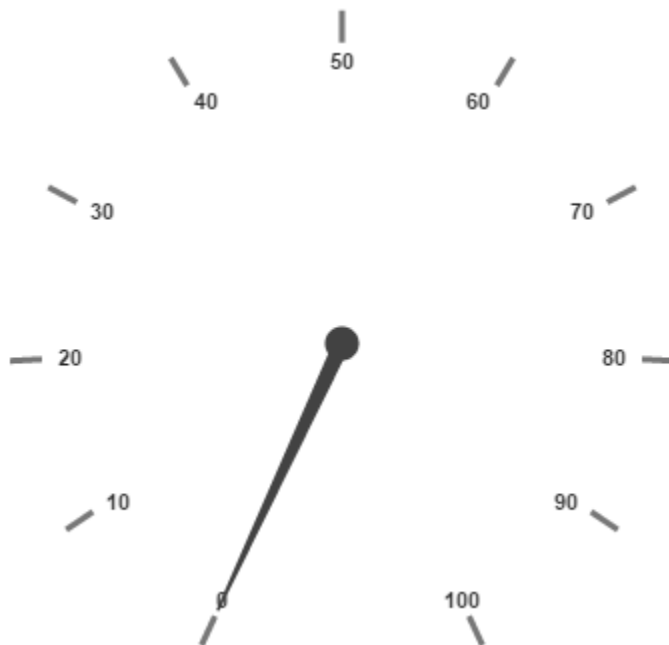


```

module CircularGaugeComponent {
  $(function () {
    //For circular gauge rendering
    var circularGaugeSample = new
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {
      scales: [{
        ticks: [{
          value: 30
        }]
      }]
    });
  });
}

```

Execute the above code to render the following output.



### Tick Customization

Height and width of the ticks can be applied by using the properties **height** and **width**. You can customize ticks with the properties such as angle, color, etc. **angle** attribute is used to display the labels in the specified angles and **color** attribute is used to display the labels in specified color. Ticks are two types such as major and minor.

Major **type** ticks are for major interval values and minor type ticks are for minor interval values. You can position ticks with the help of two properties such as **distanceFromScale** and **placement**.

**distanceFromScale** property defines the distance between the scale and ticks. **Placement** property is used to locate the ticks with respect to scale either inside the scale or outside the scale or along the scale. It is an enumerable data type.

### HTML

```

<div id="CircularGauge1"></div>

```

## JAVASCRIPT

```
$(function () {  
  // For Circular Gauge rendering  
  var circularGaugeSample = new  
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
    scales: [{  
      ticks: [  
        // For setting tick1  
        { type: "major", color: "Red" },  
        // For setting tick2  
        {  
          // For setting tick type  
          type: "minor",  
          // For setting tick color  
          color: "yellow",  
          // For setting tick height  
          height: 8,  
          // For setting tick placement  
          placement: "near",  
          // For setting tick distance from scale  
          distanceFromScale: 5}]  
      }]  
    });  
  });  
});
```

Execute the above code to render the following output.



## Indicators

Indicators simply indicates the current status of the pointer. **indicators** are in several formats such as in shape format, textual format and image format.

### Adding Indicator Collection

Indicators collection is directly added to the scale object. Refer the following code to add indicator collection in a **Gauge** control.

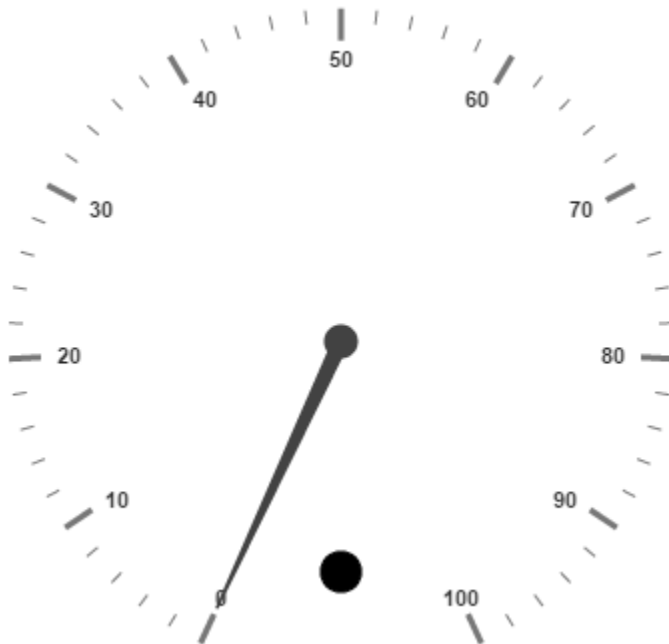
#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {  
    //For circular gauge rendering  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
        scales: [{  
            showIndicators: true,  
            indicators: [{  
                // For setting indicator height  
                height: 10,  
                // For setting indicator width  
                width: 10,  
                // For setting indicator type  
                type: "circle",  
                // For setting indicator value  
                value: 0,  
                // For setting indicator position  
                position: { x: 185, y: 300 },  
            }]  
        }]  
    });  
});
```

Execute the above code to render the following output.



### Basic Customization

You can enable indicators by setting `showIndicators` to 'true'. The `height` and `width` property for the indicators are used to specify the area allocated to the indicator for the width and height respectively. You can use the `position` collection to position the indicators along `x` and `y` axis.

Indicators are of several types such as, circle, rectangle, rounded rectangle, text and image. By using the `type` property you can avail those shapes. For image type `imageUrl` property is used.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CircularGaugeComponent {
    $(function () {
        // For Circular Gauge rendering
        var circularGaugeSample = new
        ej.datavisualization.CircularGauge($("#CircularGauge1"), {
            scales: [{
                showIndicators: true, minorIntervalValue: 5,
                backgroundColor: "#5DF243",
                border: { width: 1.5, color: "black" },
                showScaleBar: true, radius: 150, size: 5,
                pointers: [{
                    backgroundColor: "#5DF243",
                    border: { width: 1.5, color: "black" },
                    length: 110
                }],
            }],
            indicators: [{
                // For setting indicator height
                height: 10,
                // For setting indicator width
                width: 10,
                // For setting indicator type
                type: "circle",
                // For setting indicator value
                value: 0,
                // For setting indicator position
                position: { x: 185, y: 300 },
            }],
        });
    });
}
```

Execute the above code to render the following output.



### State Ranges

State Ranges are used to specify the indicator behavior in the specified region. Use `startValue` and `endValue` to set the range bound for the pointer. Whenever the pointer crosses the specified region, the indicator attributes are applied for ranges.

The `backgroundColor` and `borderColor` sets the appearance behavior for the indicators. For text type indicators you can give value for text. And `text` can be changed whenever the pointer crosses its state range area. There are many basic `font` options available for the text in the state range such as `size`, `fontStyle` and `fontFamily`.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {
    // For Circular Gauge rendering
    var circularGaugeSample = new
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {
        scales: [{
            showIndicators: true, minorIntervalValue: 5,
            backgroundColor: "#5DF243",
            border: { width: 1.5, color: "black" },
            showScaleBar: true, radius: 150, size: 5,
            pointers: [{
                backgroundColor: "#5DF243",
                border: { width: 1.5, color: "black" },
                length: 110
            }],
            indicators: [{
                // For setting indicator height
                height: 10,
                // For setting indicator width
```

```

width: 10,
// For setting indicator type
type: "circle",
// For setting indicator value
value: 0,
// For setting indicator position
position: { x: 185, y: 300 },
// For setting indicator state range collection
stateRanges: [{
// For setting state range end value height
endValue: 100,
// For setting state range start value
startValue: 0,
// For setting indicator background color
backgroundColor: "#5DF243",
// For setting indicator border color
borderColor: "Black",
// For setting indicator text
text: "",
// For setting indicator text color
textColor: "#870505"
}]
},
}
});
});

```

Execute the above code to render the following output.



### Multiple Indicators

You can use multiple indicators for a single **Gauge**. Each indicator have a list of `stateRanges`. Refer the following code example for multiple Indicators.

#### HTML

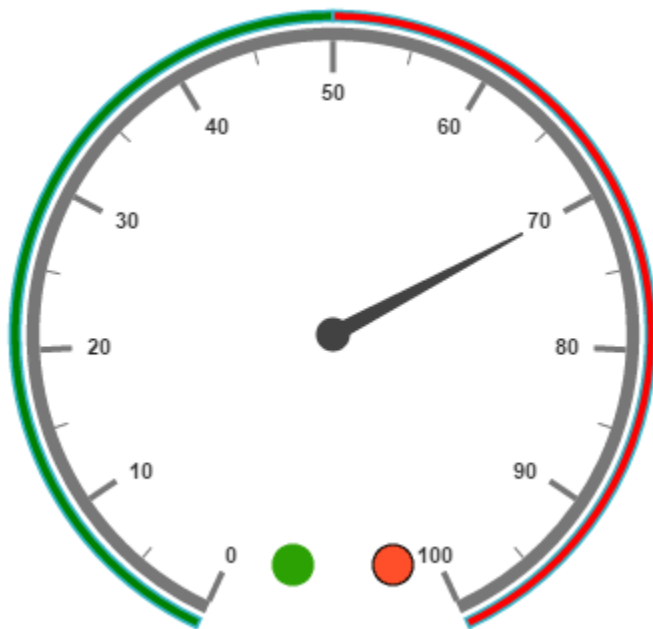
```
<div id="CircularGauge1"></div>
```

**JAVASCRIPT**

```
$(function () {  
    // For Circular Gauge rendering  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
        scales: [{  
            readOnly: false,  
            showIndicators: true, showRanges: true,  
            minorIntervalValue: 5,  
            showScaleBar: true, radius: 150, size: 5,  
            pointers: [{  
                length: 110, value: 70  
            }],  
            ranges: [{  
                startValue: 0, endValue: 50,  
                backgroundColor: "Green",  
                placement: "far", distanceFromScale: -30  
            },  
            {  
                startValue: 50, endValue: 100,  
                backgroundColor: "red",  
                placement: "far", distanceFromScale: -30  
            }],  
            indicators: [  
                //Indicator1  
                {  
                    height: 10,  
                    width: 10,  
                    type: "circle",  
                    value: 0,  
                    position: { x: 165, y: 300 },  
                    stateRanges: [{  
                        endValue: 50,  
                        startValue: 0,  
                        backgroundColor: "#24F92F",  
                        borderColor: "Black"  
                    }, {  
                        endValue: 50,  
                        startValue: 100,  
                        backgroundColor: "#322C04",  
                        borderColor: "Black"  
                    }]  
                },  
                //Indicator2  
                {  
                    height: 10,  
                    width: 10,  
                    type: "circle",  
                    value: 0,  
                    position: { x: 215, y: 300 },  
                    stateRanges: [{  
                        endValue: 50,  
                        startValue: 0,  
                        backgroundColor: "#600000",
```

```
borderColor: "Black"
}, {
  endValue: 100,
  startValue: 50,
  backgroundColor: "#FF4F2A",
  borderColor: "Black"
}]
}],
}]
});
});
```

Execute the above code to render the following output.



### Ranges and Frames

Ranges are used to specify or group the scale values. By using `ranges`, you can describe the values in the pointers.

#### Adding Range Collection

Range collection is directly added to the scale object. Refer the following code example to add range collection in a **Gauge** control.

#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CircularGaugeComponent {
  $(function () {
    //For circular gauge rendering
```



```

var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
  scales: [{
    showRanges: true,
    ranges: [{
      startValue: 20,
      endValue: 80
    }]
  }]
});

```

## Range Customization

### Appearance

The **API size** is used to specify the width of the ranges. The major attributes for ranges are **startValue** and **endValue**. **startValue** defines the start position of the ranges and **endValue** defines the end position of the ranges.

**startWidth** and **endWidth** are used to specify the range width at the starting and ending position of the ranges. You can add the gradient effects and opacity to the ranges by using **gradients** object and **opacity** property.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```

$(function () {
  // For Circular Gauge rendering
  var circularGaugeSample = new
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {
    scales: [{
      showRanges: true,
      showScaleBar: true,
      radius: 150, size: 2,
      ranges: [{
        //For setting range start value
        startValue: 20,
        //For setting range end value
        endValue: 80,
        //For setting range background color
        backgroundColor: "Green",
        //For setting range opacity
        opacity: 1
      }]
    }]
  });
});

```

Execute the above code to render the following output.



### Colors and Border

By customizing the ranges, the appearance of the **Gauge** can be improved. The range border is modified with the object called **border**. It has two border property such as **color** and **width**. These are used to customize the border color of the ranges and border width of the ranges.

You can set the background color to improve the look and feel of the **Circular Gauge**. For customizing the background color of the ranges, **backgroundColor** is used.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {
  // For Circular Gauge rendering
  var circularGaugeSample = new
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {
    scales: [{
      showRanges: true,
      showScaleBar: true,
      radius: 150, size: 2,
      ranges: [{
        //For setting range start value
        startValue: 20,
        //For setting range end value
        endValue: 80,
        //For setting range background color
        backgroundColor: "yellow",
        //For setting range border
        border: { color: "green", width: 2 },
      }]
    }]
  });
```

```
});
```

Execute the above code to render the following output.



### Positioning the ranges

You can position ranges using two properties such as `distanceFromScale` and `placement`.

**distanceFromScale** property defines the distance between the scale and range. **Placement** property is used to locate the pointer with respect to scale either inside the scale or outside the scale or along the scale. It is an enumerable data type.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {
    // For Circular Gauge rendering
    var circularGaugeSample = new
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {
        scales: [{
            showRanges: true,
            showScaleBar: true,
            radius: 150, size: 2,
            ranges: [{
                //For setting range start value
                startValue: 0,
                //For setting range end value
                endValue: 100,
                //For setting range background color
                backgroundColor: "Green",
                //For setting range placement
                placement: "far",
                //For setting distance between scale and ranges
                distanceFromScale: -30,
```

```
//For setting range border  
border: { color: "Black", width: 2 },  
}]  
}]  
});  
});
```

Execute the above code to render the following output.



### Multiple Ranges

You can set multiple ranges by adding an array of ranges objects. Refer the following code example for multiple ranges functionality.

#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {  
  // For Circular Gauge rendering  
  var circularGaugeSample = new  
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
    scales: [{  
      showRanges: true,  
      showScaleBar: true,  
      radius: 150, size: 2,  
      pointers: [{  
        value: 40,  
        showBackNeedle: true,  
        length: 100  
      }],  
      ranges: [  
        //For setting range1
```

```
{
  startValue: 0, endValue: 50,
  backgroundColor: "Green",
  placement: "far", distanceFromScale: -30
},
//For setting range2
{
  startValue: 50, endValue: 80,
  backgroundColor: "yellow",
  placement: "far", distanceFromScale: -30
},
//For setting range3
{
  startValue: 80, endValue: 100,
  backgroundColor: "red",
  placement: "far", distanceFromScale: -30
}]
}];
});
```

Execute the above code to render the following output.



### Frames

Frame is the element that decides the appearance of the **Circular Gauge**. You can customize it using the object called `frame`. It has the properties such as `frameType`, `backGroundUrl`, `halfCircleFrameStartAngle` and `halfCircleFrameEndAngle`.

`frameType` is used to specify whether frame is a half circle frame or full circle frame.

`halfCircleFrameStartAngle` and `halfCircleFrameEndAngle` are used to specify the angle for **Gauge** with frame type as half circle. `backgroundUrl` is used to set the background image for the frame.

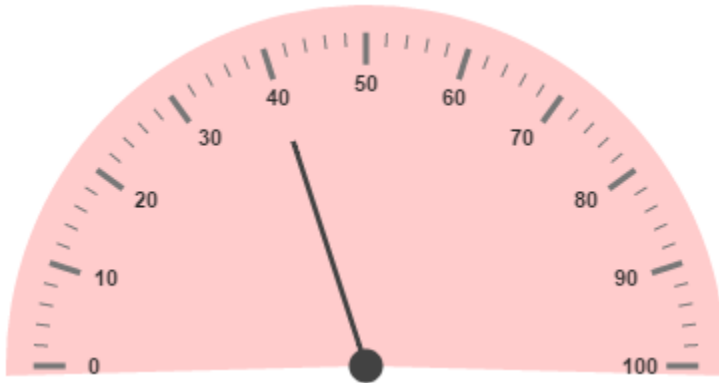
### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {
    // For Circular Gauge rendering
    var circularGaugeSample = new
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {
        frame: {
            //For setting type
            frameType: "halfcircle",
            //For setting half circle frame start angle
            halfCircleFrameStartAngle: 205,
            //For setting half circle frame end angle
            halfCircleFrameEndAngle: 335,
        }, pointerCap: { radius: 50 },
        backgroundColor: "#FFCCCC",
        scales: [{
            startAngle: 180, sweepAngle: 180,
            pointers: [{
                needleType: "rectangle",
                width: 1, length: 120, value: 40
            }]
        }]
    });
});
```

Execute the above code to render the following output.



### Legend

The **legend** contains the list of the ranges that appear in the circular gauge

### Legend Visibility

By default, the legend will not be displayed in the circular gauge. You can enable or disable it by using the **visible** property of the legend.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CircularGaugeComponent {
    $(function () {
```

```
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
  legend: {
    visible : true,
  },
});
});
}
```

Measure of wind speed in km/h



Light air    Light breeze    Gentle breeze    Moderate breeze  
 Strong breeze    Gale    Storm    Hurricane force

[Click](#) here to view the online demo sample for legend in the circular Gauge.

#### Legend Text

The text displayed in the legend can be customized by using the `legendText` property present in the ranges of the circular gauge. When the `legendText` is not specified in the ranges, then the legend item for that particular range will not displayed. By default the `legendText` value is `null`.

#### JAVASCRIPT

```
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
  // ...
  ranges: [{
    legendText: "Light air",
  }]
  //...
  // ...
});
```

### Position and Align the Legend

By using the `position` property, you can position the legend at *left*, *right*, *top* or *bottom* of the CircularGauge. The legend is positioned at the **bottom** of the circular gauge, by default.

#### JAVASCRIPT

```
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
// ...
legend: {
// ...
//Place the legend at top of the circular gauge
position: 'top',
}
// ...
});
```

Measure of wind speed in km/h

Light air   Light breeze   Gentle breeze   Moderate breeze  
Strong breeze   Gale   Storm   Hurricane force



### Legend Alignment

You can align the legend to the *center*, *far* or *near* based on its position by using the `alignment` property.

#### JAVASCRIPT

```
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
// ...
legend: {
//...
//The below two settings will place the legend at the top-right corner of
the circular gauge.
}
```



```
alignment: 'far',  
position: 'top',  
}  
// ...  
});
```

### Measure of wind speed in km/h

Light air   Light breeze   Gentle breeze   Moderate breeze  
Strong breeze   Gale   Storm   Hurricane force



### Customization

#### Legend Fill and Opacity

You can change the opacity and fill color of legend text using `opacity` and `fill` property of legend.

#### JAVASCRIPT

```
var circularGaugeSample = new  
ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
  // ...  
  legend: {  
    //...  
    //fill color of legend  
    fill: 'red',  
    //opacity of legend  
    opacity: 0.8  
  }  
  // ...  
});
```

### Legend shape

To change the legend item shape, you have to specify the desired shape in the **shape** property of the legend. By default, the shape of the legend is **circle**. It also supports rectangle, diamond, triangle, slider, line, pentagon, trapezoid and wedge shapes.

### JAVASCRIPT

```
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
// ...
legend: {
//...
//Change legend shape
shape: 'slider',
}
// ...
});
```

Measure of wind speed in km/h



■ Light air   
 ■ Light breeze   
 ■ Gentle breeze   
 ■ Moderate breeze  
■ Strong breeze   
 ■ Gale   
 ■ Storm   
 ■ Hurricane force

### Legend Item Size and Border

You can change the size of the legend items by using the **width** and **height** properties in the **itemStyle**. To change the legend item border, use **border** property of the legend itemStyle. The color and width of legend item border can be customized using **border color** and **width** property.

### JAVASCRIPT

```
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
// ...
});
```

```

legend: {
  //...
  //Change legend items border, height and width
  itemStyle: {width: 13, height: 13, border: { color: "#FF0000", width: 2 } },
}
// ...
});

```

Measure of wind speed in km/h



Light air   Light breeze   Gentle breeze   Moderate breeze  
 Strong breeze   Gale   Storm   Hurricane force

### Legend size

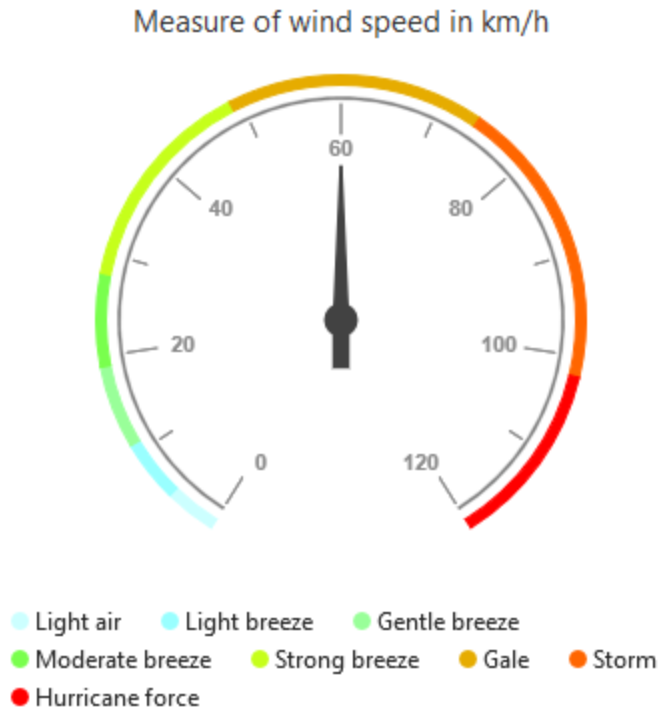
You can change the default legend size by using the `size` property of the legend. The height and width of legend size can customized using `height` and `width` property.

### JAVASCRIPT

```

var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
  // ...
  legend: {
    //...
    //Change legend size
    size:{width: '350', height: '100'}
  }
  // ...
});

```

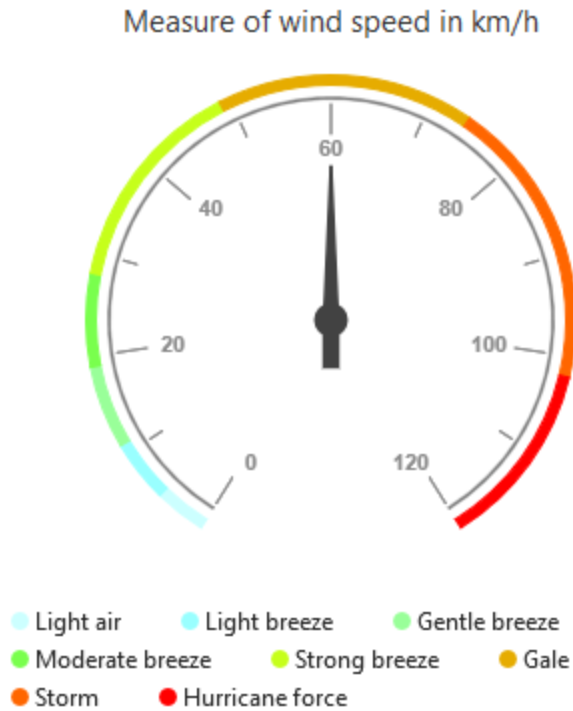


#### Legend Item Padding

You can control the spacing between the legend items by using the `itemPadding` option of the legend. The default value is 20.

#### JAVASCRIPT

```
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
// ...
legend: {
//...
//Add space between each legend item
itemPadding: 30,
}
// ...
});
```

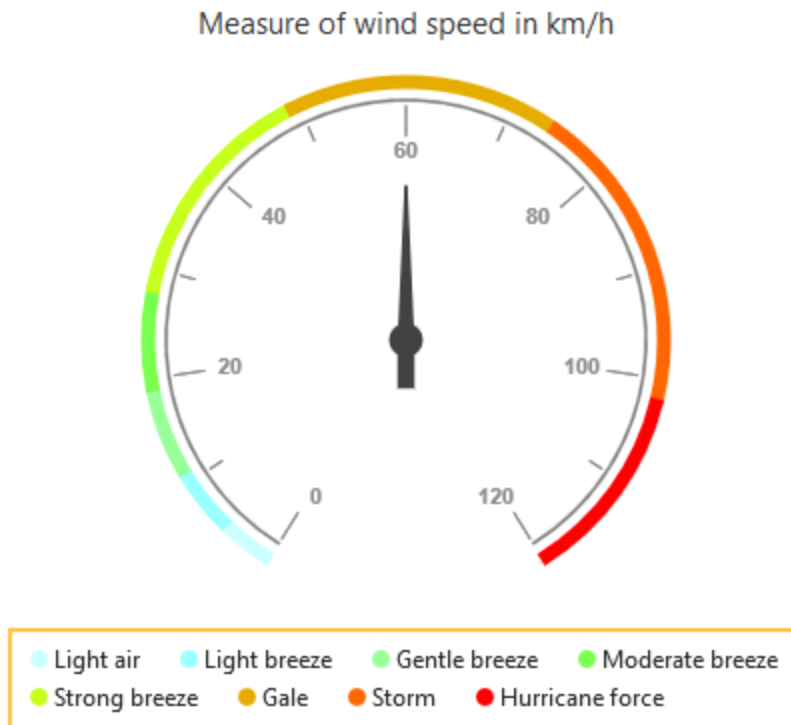


#### Legend border

You can customize the legend border by using the `border` option in the legend. The legend border can be customized using `border color` and `width` property.

#### JAVASCRIPT

```
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
// ...
legend: {
//...
//Set border color and width to legend
border: {color: "#FFC342", width: 2},
}
// ...
});
```

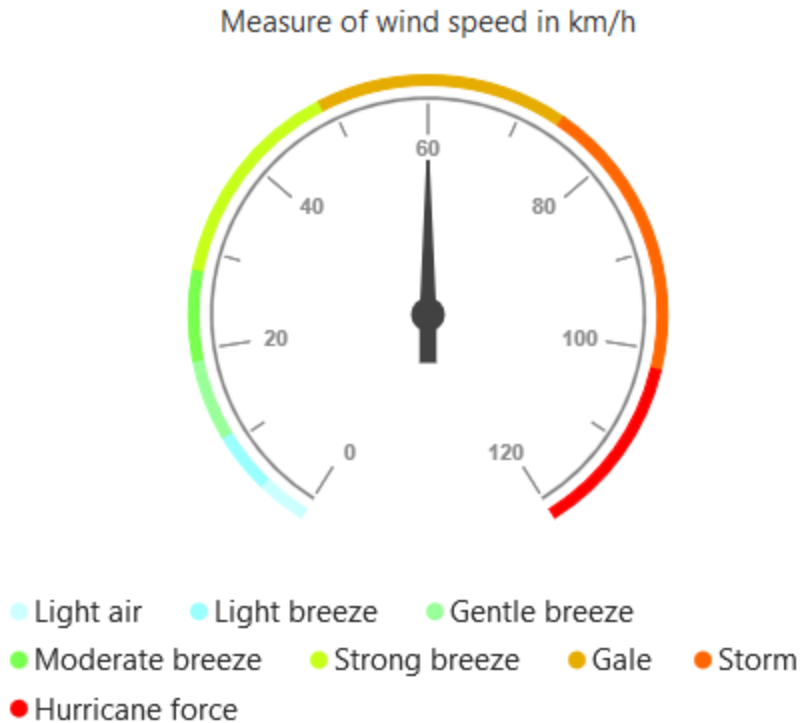


#### Font of the legend text

The font of the legend item text can be customized by using properties such as `fontFamily`, `fontStyle`, `fontWeight` and `size` of legend font.

#### JAVASCRIPT

```
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
// ...
legend: {
//...
//Customize the legend item text
font: { fontFamily: 'Segoe UI', fontStyle: 'Normal', fontWeight: 'Bold',
size: '15px' },
},
// ...
});
```



## Events

### Legend Item Render

**LegendItemRender** event triggers before rendering the legend items. This event is triggered for each legend item in Circular gauge. You can use this event to customize legend item shape or add custom text to legend item dynamically

### JAVASCRIPT

```
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
  //Subscribe the legendItemRender event
  legendItemRender: "onLegendRender",
  //...
});
function onLegendRender(sender) {
  //Get legend item details before rendering
  var legendItem = sender.data;
}
```

### Legend Item Click

You can get the legend item details such as *RangeIndex*, *bounds* and *shape* by subscribing the **legendItemClick** event of the circular gauge. When the legend item is clicked, it triggers the event and returns the legend information

### JAVASCRIPT

```
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
  //Subscribe the legend item click event
```

```

legendItemClick: "onLegendClicked",
//...
});
function onLegendClicked(sender) {
//Get legend item details on legend item click.
var legendItem = sender.data;
}

```

## Custom labels

Custom labels are the texts that you can use them in any location of the **Gauge**.

### Adding Custom Label Collection

Custom labels collection is directly added to the scale object. Refer the following code to add **customLabels** collection in a **Gauge** control.

#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CircularGaugeComponent {
$(function () {
//For circular gauge rendering
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
scales: [{
showCustomLabels: true,
customLabels: [{
color: "Red"
}]
}]
})
});
}

```

### Basic Customization

You can customize custom labels using the properties such as **textAngle**, **color** and **font**. **textAngle** attribute is used to display the custom labels in the specified angles and **color** attribute is used to display the custom labels in specified color.

You can use **value** attribute to set the text value in the custom labels. To display the custom labels, set **showCustomLabels** as 'true'. To set the location of the custom label in **Circular Gauge**, **position** property is used. By using **x** and **y** axis you can adjust the position of the custom labels.

Font option is also available on custom labels. The basic three properties of fonts such as size, family and style can be achieved by **size**, **fontStyle** and **fontFamily** attributes.

#### HTML

```
<div id="CircularGauge1"></div>
```



**JAVASCRIPT**

```
$(function () {  
  // For Circular Gauge rendering  
  var circularGaugeSample = new  
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
    scales: [{  
      size: 10,  
      shadowOffset: 10,  
      showCustomLabels: true,  
      showRanges: true,  
      showScaleBar: true,  
      radius: 150, size: 2,  
      customLabels: [{  
        //For setting custom label text angle  
        textAngle: 10,  
        //For setting custom label color  
        color: "Red",  
        //For setting custom label value  
        value: "CustomLabel1",  
        //For setting custom label font option  
        font: {  
          size: "18px",  
          fontFamily: "Arial",  
          fontStyle: "bold"  
        },  
        position: { x: 180, y: 100 }  
      }]  
    }]  
  });  
});
```

Execute the above code to render the following output.



### Multiple Custom Labels

You can set multiple custom labels in a single **Circular Gauge** by adding an array of custom label objects. Refer the following code example for multiple custom label functionality.

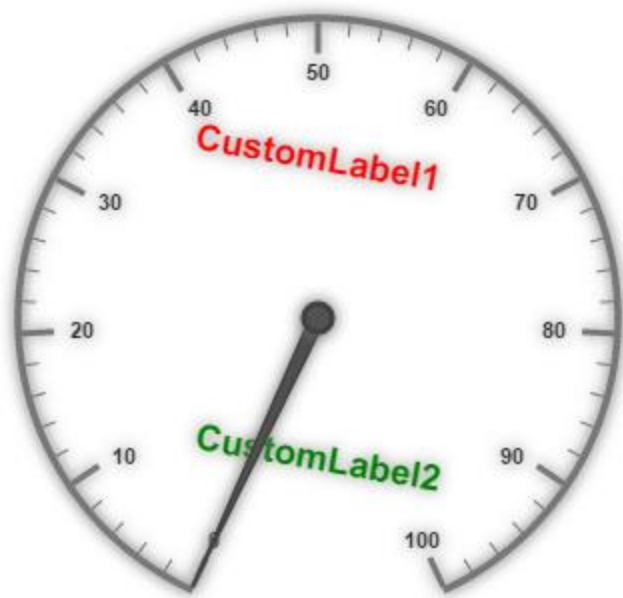
#### HTML

```
<div id="CircularGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {  
  // For Circular Gauge rendering  
  var circularGaugeSample = new  
  ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
    scales: [{  
      size: 10,  
      shadowOffset: 10,  
      showCustomLabels: true,  
      showRanges: true,  
      showScaleBar: true,  
      radius: 150, size: 2,  
      customLabels: [  
        //custom label1  
        {  
          textAngle: 10,  
          color: "Red",  
          value: "CustomLabel1",  
          font: {  
            size: "18px",  
            fontFamily: "Arial",  
            fontStyle: "bold"  
          },  
          position: { x: 180, y: 100 }  
        },  
        //custom label2  
        {  
          textAngle: 10,  
          color: "Green",  
          value: "CustomLabel2",  
          font: {  
            size: "18px",  
            fontFamily: "Arial",  
            fontStyle: "bold"  
          },  
          position: { x: 180, y: 250 }  
        }  
      ]  
    }]  
  });  
});
```

Execute the above code to render the following output.



### Outer Custom Label

**Outer Custom Label** is used to show custom labels outside the **gauge** control. The **Outer Custom Label** can be positioned with API called `outerCustomLabelPosition`. The value for this API is enumerable type and its possible values are,

- Right
- Left
- Top
- Bottom

When a custom label is to be displayed as an **Outer Custom Label**, set the API `positionType` as `Outer`. Refer to the following code example to get the **Outer Custom Label**.

### HTML

```
<div id="CircularGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    var circularGaugeSample = new  
    ej.datavisualization.CircularGauge($("#CircularGauge1"), {  
        // Sets outer custom label position.  
        outerCustomLabelPosition: "right",  
        //Defines the tooltip object.  
        tooltip: {  
            // Enables the custom label tooltip.  
            showCustomLabelTooltip: true,  
        },  
        // Customizes the scale options.  
        scales: [{  
            showLabels: true,  
            radius: 150,  
            // Customizes the custom label options.
```

```

customLabels: [{
  value: "Average Speed",
  position: { x: 360, y: 30 },
  color: "Red",
  font: {
    size: "18px",
    fontFamily: "Arial",
    fontStyle: "bold"
  },
  positionType: "outer",
}],
// Customizes the pointers options.
pointers: [{
  value: 60,
  length: 100,
}]
});

```

Execute the above code to render the following output.



### Tooltip

**Tooltip** feature has been added to the **Circular Gauge**. **Circular Gauge** has several elements such as pointers, label, customLabel, scales, etc. There is a need for **Tooltip** feature in the **Circular Gauge** control because whenever the text hides or overrides with other gauge elements, it may not be fully visible. For resolving those problems **tooltip** feature has been implemented in the **Circular Gauge** control.

### Default Tooltip

**Tooltip** has three attributes in it. The first two attributes such as **showLabelText** and **showCustomLabelText** are for enabling the **Tooltip** for label as well as custom label in default appearance.

**ShowLabelText** is to enable the **Tooltip** for labels and **showCustomLabelText** is for enabling the **Tooltip** option for customLabels.

## HTML

```
<div id="tooltipGauge"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CircularGaugeComponent {
$(function () {
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#tooltipGauge"), {
//Defines the tooltip object.
tooltip: {
//Enables the label tooltip.
showLabelTooltip: true,
//Enables the custom label tooltip.
showCustomLabelTooltip: true,
},
//Customizes the scale options.
scales: [{
showLabels: true,
radius: 150,
//Customizes the custom label options.
customLabels: [{
value: "095345",
font: {
size: "18px",
fontFamily: "Arial",
fontStyle: "bold"
},
position: { x: 180, y: 220 }
}],
//Customizes the pointers options.
pointers: [{
value: 60,
length: 100,
}]
}];
});
}
```

Execute the above code to render the following output.



### Tooltip Template

In **Tooltip** option, you can customize the Tooltip window by adding the tooltip template on that page with the help of API `templateID`. Refer to the following code example to know more about Tooltip template.

#### HTML

```
<div id="Tooltip" style="height: 60px; display: none;">
<div id="icon">
<div id="eficon"></div>
</div>
<div id="value">
<div>
<label id="efpercentage">&#160;#label#</label>
</div>
</div>
</div>
<div id="tooltipGauge"></div>
```

#### JAVASCRIPT

```
$(function () {
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#tooltipGauge"), {
//Defines the tooltip object.
tooltip: {
// Enables the label tooltip.
showLabelTooltip: true,
// Enables the custom label tooltip.
showCustomLabelTooltip: true,
// Adds tooltip template.
templateID: "Tooltip"
},
// Customizes the scale options.
scales: [{
showLabels: true,
```

```
radius: 150,
// Customizes the custom label options.
customLabels: [{
value: "0 9 5 3 4 5",
font: {
size: "18px",
fontFamily: "Arial",
fontStyle: "bold"
},
position: { x: 180, y: 220 }
}],
// Customizes the pointers options.
pointers: [{
value: 60,
length: 100,
}]
}];
});
```

### HTML

```
<style type="text/css">
<!-- Adds the necessary styles here. -->.
</style>
```

Execute the above code to render the following output.



### Sub Gauges

A **Circular Gauge** containing another circular gauge is said to be **subGauges**. In order to make a sample like watch that has second gauge, minute gauge and hour gauge, sub gauges are used.

#### Adding SubGauges

Sub gauge collection is directly added to the scale object. Refer the following code example to add custom sub gauge collection in a **Gauge** control.

**HTML**

```
<div id="CircularGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CircularGaugeComponent {
$(function () {
//For circular gauge rendering
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
scales: [{ showSubGauges: true,
subGauges: [{
gaugeID: "Gauge1"
}]
}]
});
}
```

**Basic Customization**

Basic attributes such as **height** and **width** property are used to set height and width of the sub gauge. You can easily position the gauge in another gauge using the **position** object and by giving the **x** and **y** Coordinates value. **controlID** attribute is used to specify the sub gauge ID.

**HTML**

```
<div id=" SubGauge1"></div>
<div id="CircularGauge1"> </div>
```

**JAVASCRIPT**

```
$(function () {
var circularGauge = new ej.datavisualization.CircularGauge($("#SubGauge1"), {
backgroundColor: "Blue",value:50, radius: 110,
scales: [{
radius: 110,
}]
});
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
height: 500,value:50, width: 500,
scales: [{radius:190,
subGauges:[{
//For setting sub gauge control ID
controlID: "SubGauge1",
//For setting sub gauge height
height:250,
//For setting sub gauge width
width: 250,
//For setting sub gauge position
position: { x: 150, y: 100 }
}]
}]
});
}
```



```

    }]
  }]
});
});

```

Execute the above code to render the following output.



### Multiple SubGauges

You can set multiple sub gauges in a single **Circular Gauge** by adding an array of sub gauge objects. Refer the following code example for multiple sub gauges functionality.

#### HTML

```

<div id="CircularGauge1"></div>
<div id=" SubGauge1"> </div>
<div id=" SubGauge2"> </div>

```

#### JAVASCRIPT

```

$(function () {
  var circular = new ej.datavisualization.CircularGauge($("#SubGauge1"), {
    backgroundColor: "#f5b43f",
    scales: [{
      radius: 150
    }]
  });
  var circularGauge = new ej.datavisualization.CircularGauge($("#SubGauge2"), {
    backgroundColor: "#f5b43f",
    scales: [{
      radius: 150
    }]
  });
});

```

```
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
height: 500,
width: 500,
scales: [{
radius: 250,
subGauges: [
//Sub gauge1
{
controlID: "SubGauge1",
height: 200,
width: 200,
position: { x: 200, y: 150 }
},
//Sub gauge2
{
controlID: "SubGauge2",
height: 200,
width: 200,
position: { x: 50, y: 200 }
}]
}]
});
```

Execute the above code to render the following output.



### Gauge Position

**Semi-circular Gauge** can be positioned within the canvas element which provides better appearance for the gauge in the canvas.

#### Positioning

Semi-circular Gauge can be positioned with the help of the attribute called `gaugePosition`. It is an enumerable value. You can position the gauge away from the corner with the help of the `distanceFromCorner` attribute.

The possible enum values for the `gaugePosition` are as follows:

- Top left
- Top center
- Top right
- Middle left
- Center
- Middle right
- Bottom left
- Bottom center
- Bottom right

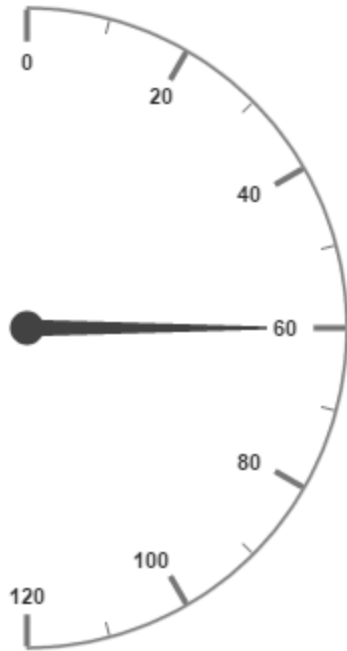
## HTML

```
<div style="float: left" id="gauge1"></div>
<div id=" CoreCircularGauge "></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CircularGaugeComponent {
$(function() {
var circularGaugeSample = new
ej.datavisualization.CircularGauge($("#CircularGauge1"), {
backgroundColor: "transparent",
// To set dimension of the canvas.
width: 800,
height: 500,
// To set the value and radius of the canvas frame.
radius: 120,
value: 60,
// To set the gauge position.
gaugePosition: "center",
// To set the distance from the corner.
distanceFromCorner: 30,
// To set the semicircle frame specifications.
frame: {
frameType: 'halfcircle',
halfCircleFrameStartAngle: 270,
halfCircleFrameEndAngle: 90
},
// To set the scale specification.
scales: [{
startAngle: 270,
sweepAngle: 180,
radius: 160,
showScaleBar: true,
size: 1,
maximum: 120,
majorIntervalValue: 20,
minorIntervalValue: 10,
border: {
width: 0.5,
}
}
}];
});
});
}
```

Execute the above code to render the following output.



## Exporting

**Circular Gauge** has an exporting feature that converts **Gauge** control into image format and then export in client side. The method API `exportImage` is used to export the **Circular Gauge**. It has two arguments such as **file name** and **file format** to specify the file name and file formats. For exporting refer the following code example.

### HTML

```
<input type="submit" value="Export Image" id="buttonExportImage">
<div id="circularGauge "></div>
<div id="txtFileName">FileName </div>
<div id="ddFileType">FileFormat </div>
</input>
<select id="Select1">
<option value="JPEG">JPEG</option>
<option value="PNG">PNG</option>
</select>
```

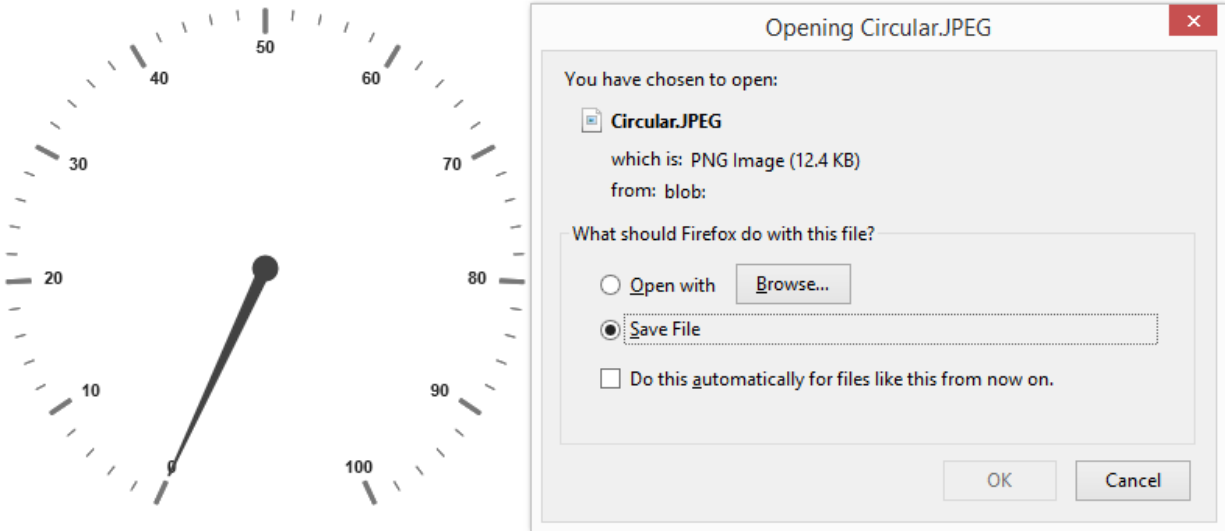
### JAVASCRIPT

```
$(function () {
    var circularGaugeSample = new
    ej.datavisualization.CircularGauge($("#circularGauge"));
    var basicButton = new ej.Button($("#buttonExportImage"), {
        width: "100px", click: "buttonClickEvent",
    });
});
function buttonClickEvent() {
    var FileName = $("#txtFileName").val();
    var FileFormat = $("#ddFileType").val();
    var circularGaugeSample = new
    ej.datavisualization.CircularGauge($("#circularGauge"));
    circularGaugeSample.exportImage(FileName, FileFormat);
}
```

```
}

```

Execute the above code to render the following output.



## Methods and Events

### Public Methods

#### *Destroying the Circular Gauge*

The `destroy` method is used to destroy the **CircularGauge** widget. All events bound using this.\_on will be unbind automatically and bring the control to pre-init state.

### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
///...///
});
// destroy the CircularGauge
sample.destroy();
});
});
}
```

#### *Getting Back Needle Length*

The `getBackNeedleLength` method is used to get the needle length of **CircularGauge**.

### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
```

```

var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.getBackNeedleLength();
});
});
}

```

### Getting Custom Label Angle

The `getCustomLabelAngle` method is used to get the angle of custom label.

#### JAVASCRIPT

```

<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.getCustomLabelAngle();
});
});
}

```

### Getting Custom Label value

The `getCustomLabelValue` method is used to get the value of custom label.

#### JAVASCRIPT

```

<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.getCustomLabelValue();
});
});
}

```

### Getting Label Angle

The `getLabelAngle` method is used to get angle of label.

#### JAVASCRIPT

```

<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
});
}

```

```
sample.getLabelAngle();
});
});
}
```

### Getting Label Distance From Scale

The `getLabelDistanceFromScale` method is used to get the distance value from scale for label.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.getLabelDistanceFromScale();
});
});
}
```

### Getting Label Placement

The `getLabelPlacement` method is used to get placement of label.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.getLabelPlacement();
});
});
}
```

### Getting Label Style

The `getLabelStyle` method is used to get style of label.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.getLabelStyle();
});
});
}
```



### Getting Major Interval Value

The `getMajorIntervalValue` method is used to get major interval value of CircularGauge.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getMajorIntervalValue();
  });
};
```

### Getting Marker Distance From Scale

The `getMarkerDistanceFromScale` method is used to get distance from scale value of marker.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getMarkerDistanceFromScale();
  });
};
```

### Getting Marker Style

The `getMarkerStyle` method is used to get distance from scale value of marker.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getMarkerStyle();
  });
};
```

### Getting Maximum Value

The `getMaximumValue` method is used to get maximum value of CircularGauge.

**JAVASCRIPT**

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getMaximumValue();
  });
}
```

*Getting Minimum Value*

The `getMinimumValue` method is used to get minimum value of CircularGauge.

**JAVASCRIPT**

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getMinimumValue();
  });
}
```

*Getting Minor Interval Value*

The `getMinorIntervalValue` method is used to get minor interval value of CircularGauge.

**JAVASCRIPT**

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getMinorIntervalValue();
  });
}
```

*Getting Needle Style*

The `getNeedleStyle` method is used to get style of needle.

**JAVASCRIPT**

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
```

```
$(function () {
  var sample = new
  ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
    //...//
  });
  sample.getNeedleStyle();
});
});
}
```

#### Getting Pointer Cap Border Width

The `getPointerCapBorderWidth` method is used to get border width of pointer cap.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getPointerCapBorderWidth();
  });
});
}
```

#### Getting Pointer Cap Radius

The `getPointerCapRadius` method is used to get radius of pointer cap.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getPointerCapRadius();
  });
});
}
```

#### Getting Pointer Length

The `getPointerLength` method is used to get pointer length.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
  });
});
}
```

```
});  
sample.getPointerLength();  
});  
});  
}
```

#### Getting Pointer Needle Type

The `getPointerNeedleType` method is used to get needle type of pointer.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.getPointerNeedleType();  
  });  
});  
}
```

#### Getting Pointer Placement

The `getPointerPlacement` method is used to get placement of pointer.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.getPointerPlacement();  
  });  
});  
}
```

#### Getting Pointer Value

The `getPointerValue` method is used to get pointer value.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.getPointerValue();  
  });  
});  
}
```

```
}
```

### Getting Pointer Value

The `getPointerWidth` method is used to get pointer width.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getPointerWidth();
  });
};
```

### Getting Range Border Width

The `getRangeBorderWidth` method is used to get border width of range.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getRangeBorderWidth();
  });
};
```

### Getting Range Distance From Scale

The `getRangeDistanceFromScale` method is used to get range distance from scale.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getRangeDistanceFromScale();
  });
};
```

### Getting Range End Value

The `getRangeEndValue` method is used to get end value of range.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getRangeEndValue();
  });
};
```

### Getting Range Position

The `getRangePosition` method is used to get range position.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getRangePosition();
  });
};
```

### Getting Range Size

The `getRangeSize` method is used to get range size.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getRangeSize();
  });
};
```

### Getting Range Start Value

The `getRangeStartValue` method is used to get range start value.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getRangeStartValue();
  });
};
```

#### Getting Scale Bar Size

The `getScaleBarSize` method is used to get scale bar size.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getScaleBarSize();
  });
};
```

#### Getting Scale Border Width

The `getScaleBorderWidth` method is used to get border width of scale.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getScaleBorderWidth();
  });
};
```

#### Getting Scale Direction

The `getScaleDirection` method is used to get scale direction.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
```

```
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.getScaleDirection();
});
});
}
```

#### Getting Scale Radius

The `getScaleRadius` method is used to get scale radius.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.getScaleRadius();
});
});
}
```

#### Getting Start Angle

The `getStartAngle` method is used to get start angle.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.getStartAngle();
});
});
}
```

#### Getting Sub Gauge Location

The `getSubGaugeLocation` method is used to get location of subGauge.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
});
}
```



```
sample.getSubGaugeLocation();
});
});
}
```

### Getting Sweep Angle

The `getSweepAngle` method is used to get sweep angle.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.getSweepAngle();
});
});
}
```

### Getting Tick Angle

The `getTickAngle` method is used to get tick angle.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.getTickAngle();
});
});
}
```

### Getting Tick Distance From Scale

The `getTickDistanceFromScale` method is used to get tick distance from scale value.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.getTickDistanceFromScale();
});
});
}
```

### Getting Tick Height

The `getTickHeight` method is used to get tick height.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getTickHeight();
  });
};
```

### Getting Tick Placement

The `getTickPlacement` method is used to get tick placement.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getTickPlacement();
  });
};
```

### Getting Tick Style

The `getTickStyle` method is used to get tick style.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.getTickStyle();
  });
};
```

### Getting Tick Width

The `getTickWidth` method is used to get tick width.

**JAVASCRIPT**

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.getTickWidth();
});
});
}
```

*Setting IncludeFirstValue*

The `includeFirstValue` method is used to set includeFirstValue.

**JAVASCRIPT**

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.includeFirstValue();
});
});
}
```

*Redrawing Circular Gauge*

The `redraw` method is used to redraw the Circular Gauge widget.

**JAVASCRIPT**

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.redraw();
});
});
}
```

*Setting Back Needle Length*

The `setBackNeedleLength` method is used to set the needle length of **CircularGauge**.

**JAVASCRIPT**

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
```

```
$(function () {
  var sample = new
  ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
    //...//
  });
  sample.setBackNeedleLength();
});
});
}
```

### Setting Custom Label Angle

The `setCustomLabelAngle` method is used to set the angle of custom label.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setCustomLabelAngle();
  });
});
}
```

### Setting Custom Label value

The `setCustomLabelValue` method is used to set the value of custom label.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setCustomLabelValue();
  });
});
}
```

### Setting Label Angle

The `setLabelAngle` method is used to set angle of label.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
  });
});
}
```

```
});  
sample.setLabelAngle();  
});  
});  
}
```

### Setting Label Distance From Scale

The `setLabelDistanceFromScale` method is used to set the distance value from scale for label.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.setLabelDistanceFromScale();  
  });  
});  
}
```

### Setting Label Placement

The `setLabelPlacement` method is used to set placement of label.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.setLabelPlacement();  
  });  
});  
}
```

### Setting Label Style

The `setLabelStyle` method is used to set style of label.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.setLabelStyle();  
  });  
});  
}
```

```
}
```

### Setting Major Interval Value

The `setMajorIntervalValue` method is used to set major interval value of CircularGauge.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setMajorIntervalValue();
  });
};
```

### Setting Marker Distance From Scale

The `setMarkerDistanceFromScale` method is used to set distance from scale value of marker.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setMarkerDistanceFromScale();
  });
};
```

### Setting Marker Style

The `setMarkerStyle` method is used to set distance from scale value of marker.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setMarkerStyle();
  });
};
```

### Setting Maximum Value

The `setMaximumValue` method is used to set maximum value of CircularGauge.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setMaximumValue();
  });
};
```

### Setting Minimum Value

The `setMinimumValue` method is used to set minimum value of CircularGauge.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setMinimumValue();
  });
};
```

### Setting Minor Interval Value

The `setMinorIntervalValue` method is used to set minor interval value of CircularGauge.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setMinorIntervalValue();
  });
};
```

### Setting Needle Style

The `setNeedleStyle` method is used to set style of needle.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setNeedleStyle();
  });
};
```

#### Setting Pointer Cap Border Width

The `setPointerCapBorderWidth` method is used to set border width of pointer cap.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setPointerCapBorderWidth();
  });
};
```

#### Setting Pointer Cap Radius

The `setPointerCapRadius` method is used to set radius of pointer cap.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setPointerCapRadius();
  });
};
```

#### Setting Pointer Length

The `setPointerLength` method is used to set pointer length.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
```



```
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.setPointerLength();
});
});
}
```

#### Setting Pointer Needle Type

The `setPointerNeedleType` method is used to set needle type of pointer.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.setPointerNeedleType();
});
});
}
```

#### Setting Pointer Placement

The `setPointerPlacement` method is used to set placement of pointer.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
sample.setPointerPlacement();
});
});
}
```

#### Setting Pointer Value

The `setPointerValue` method is used to set pointer value.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
$(function () {
var sample = new
ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
//...//
});
});
}
```

```
sample.setPointerValue();  
});  
});  
}
```

#### Setting Pointer Value

The `setPointerWidth` method is used to set pointer width.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.setPointerWidth();  
  });  
});  
}
```

#### Setting Range Border Width

The `setRangeBorderWidth` method is used to set border width of range.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.setRangeBorderWidth();  
  });  
});  
}
```

#### Setting Range Distance From Scale

The `setRangeDistanceFromScale` method is used to set range distance from scale.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.setRangeDistanceFromScale();  
  });  
});  
}
```

### Setting Range End Value

The `setRangeEndValue` method is used to set end value of range.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setRangeEndValue();
  });
};
```

### Setting Range Position

The `setRangePosition` method is used to set range position.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setRangePosition();
  });
};
```

### Setting Range Size

The `setRangeSize` method is used to set range size.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setRangeSize();
  });
};
```

### Setting Range Start Value

The `setRangeStartValue` method is used to set range start value.

**JAVASCRIPT**

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setRangeStartValue();
  });
}
```

*Setting Scale Bar Size*

The **setScaleBarSize** method is used to set scale bar size.

**JAVASCRIPT**

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setScaleBarSize();
  });
}
```

*Setting Scale Border Width*

The **setScaleBorderWidth** method is used to set border width of scale.

**JAVASCRIPT**

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setScaleBorderWidth();
  });
}
```

*Setting Scale Direction*

The **setScaleDirection** method is used to set scale direction.

**JAVASCRIPT**

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
```

```
$(function () {  
  var sample = new  
  ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
    //...//  
  });  
  sample.setScaleDirection();  
});  
});  
}
```

### Setting Scale Radius

The `setScaleRadius` method is used to set scale radius.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.setScaleRadius();  
  });  
});  
}
```

### Setting Start Angle

The `setStartAngle` method is used to set start angle.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.setStartAngle();  
  });  
});  
}
```

### Setting Sub Gauge Location

The `setSubGaugeLocation` method is used to set location of subGauge.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
  });  
});  
}
```

```
});  
sample.setSubGaugeLocation();  
});  
});  
}
```

### Setting Sweep Angle

The `setSweepAngle` method is used to set sweep angle.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.setSweepAngle();  
  });  
});  
}
```

### Setting Tick Angle

The `setTickAngle` method is used to set tick angle.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.setTickAngle();  
  });  
});  
}
```

### Setting Tick Distance From Scale

The `setTickDistanceFromScale` method is used to set tick distance from scale value.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>  
module CircularGaugeComponent {  
  $(function () {  
    var sample = new  
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {  
      //...//  
    });  
    sample.setTickDistanceFromScale();  
  });  
});  
}
```

```
}
```

### Setting Tick Height

The `setTickHeight` method is used to set tick height.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setTickHeight();
  });
};
```

### Setting Tick Placement

The `setTickPlacement` method is used to set tick placement.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setTickPlacement();
  });
};
```

### Setting Tick Style

The `setTickStyle` method is used to set tick style.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setTickStyle();
  });
};
```

### Setting Tick Width

The `setTickWidth` method is used to set tick width.

#### JAVASCRIPT

```
<div id="CoreCircularGauge"></div>
module CircularGaugeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.CircularGauge($("#CoreCircularGauge"), {
      //...//
    });
    sample.setTickWidth();
  });
};
```

### Events

#### Draw Custom Label

The `drawCustomLabel` event is triggered while custom labels are drawn on the gauge.

#### JAVASCRIPT

```
<script>
//drawCustomLabel event for circular gauge
$(function () {
  var sample = new ej.datavisualization.CircularGauge($("#gauge"), {
    drawCustomLabel: function () {
      //...//
    }
  });
});
</script>
```

#### Draw Indicators

The `drawIndicators` event is triggered while indicators are being drawn on the gauge.

#### JAVASCRIPT

```
<script>
//drawIndicators event for circular gauge
$(function () {
  var sample = new ej.datavisualization.CircularGauge($("#gauge"), {
    drawIndicators: function () {
      //...//
    }
  });
});
</script>
```

#### Draw Labels

The `drawLabels` event is triggered while labels are being drawn on the gauge.

#### JAVASCRIPT



```
<script>
//drawLabels event for circular gauge
$(function () {
var sample = new ej.datavisualization.CircularGauge($("#gauge"), {
drawLabels: function () {
//...//
}
});
});
</script>
```

#### Draw Pointer Cap

The **drawPointerCap** event is triggered while pointer cap is being drawn on the gauge.

#### JAVASCRIPT

```
<script>
//drawPointerCap event for circular gauge
$(function () {
var sample = new ej.datavisualization.CircularGauge($("#gauge"), {
drawPointerCap: function () {
//...//
}
});
});
</script>
```

#### Draw Pointers

The **drawPointers** event is triggered while pointer cap is being drawn on the gauge.

#### JAVASCRIPT

```
<script>
//drawPointers event for circular gauge
$(function () {
var sample = new ej.datavisualization.CircularGauge($("#gauge"), {
drawPointers: function () {
//...//
}
});
});
</script>
```

#### Draw Range

The **drawRange** event is triggered when ranges starts to be drawn on the gauge.

#### JAVASCRIPT

```
<script>
//drawRange event for circular gauge
$(function () {
var sample = new ej.datavisualization.CircularGauge($("#gauge"), {
drawRange: function () {
//...//
}
});
});
```

```
});  
});  
</script>
```

### Draw Ticks

The **drawTicks** event is triggered when ticks are being drawn on the gauge.

#### JAVASCRIPT

```
<script>  
//drawTicks event for circular gauge  
$(function () {  
  var sample = new ej.datavisualization.CircularGauge($("#gauge"), {  
    drawTicks: function () {  
      //...//  
    }  
  });  
});  
</script>
```

### Load

The **load** event is triggered when gauge starts to load.

#### JAVASCRIPT

```
<script>  
//load event for circular gauge  
$(function () {  
  var sample = new ej.datavisualization.CircularGauge($("#gauge"), {  
    load: function () {  
      //...//  
    }  
  });  
});  
</script>
```

### Mouse Click

The **mouseClick** event is triggered when left mouse button is clicked.

#### JAVASCRIPT

```
<script>  
//mouseClick event for circular gauge  
$(function () {  
  var sample = new ej.datavisualization.CircularGauge($("#gauge"), {  
    mouseClick: function () {  
      //...//  
    }  
  });  
});  
</script>
```

### Mouse Click Move

The `mouseClickMove` event is triggered when clicking and dragging the mouse pointer over the gauge pointer.

#### JAVASCRIPT

```
<script>
//mouseClickMove event for circular gauge
$(function () {
var sample = new ej.datavisualization.CircularGauge($("#gauge"), {
mouseClickMove: function () {
//...//
}
});
});
</script>
```

### Mouse Click Up

The `mouseClickUp` event is triggered when clicking and dragging the mouse pointer over the gauge pointer.

#### JAVASCRIPT

```
<script>
//mouseClickUp event for circular gauge
$(function () {
var sample = new ej.datavisualization.CircularGauge($("#gauge"), {
mouseClickUp: function () {
//...//
}
});
});
</script>
```

### Render Complete

The `renderComplete` event is triggered when rendering of the gauge is completed.

#### JAVASCRIPT

```
<script>
//renderComplete event for circular gauge
$(function () {
var sample = new ej.datavisualization.CircularGauge($("#gauge"), {
renderComplete: function () {
//...//
}
});
});
</script>
```

### Range Mouse Move

The `rangeMouseMove` event is triggered when moving mouse on ranges.

#### JAVASCRIPT

```

<script>
//rangeMouseMove event for circular gauge
$(function () {
var sample = new ej.datavisualization.CircularGauge($("#gauge"), {
rangeMouseMove: function () {
//...//
}
});
});
});
</script>

```

## MVVM

### AngularJS

Circular Gauge contains AngularJS support. You can add object as well as array object in the Circular Gauge. The two way binding support is given to the pointer value, minimum scale value and maximum scale value.

### Rendering the Circular Gauge

**ej-CircularGauge** is the control tag in which **ej** is tag prefix and **CircularGauge** is the control name. The following code example helps you to render **Circular Gauge**.

### HTML

```

<!--To Render the Circular gauge-->
<!doctype html>
<html ng-app="syncApp">
<head>
<!--Refer the necessary script here-->
</head>
<body ng-controller="CircularGauge">
<ej-circulargauge id="CircularGauge1" e-backgroundcolor="transparent" e-
value="50"
e-width="500" e-readonly="false" e-load="loadGaugeTheme"
e-enableanimation="false">
</ej-circulargauge>
<script type="text/javascript">
<!--binding the value to the scope variables in application controller-->
angular.module('syncApp', ['ejangular'])
.controller('CircularGauge', function ($scope) {
$scope.number = 50;
$scope.minimum = 0;
$scope.maximum = 120;
});
</script>
</body>
</html>

```

Execute the above code to render the output as follows.



Value

#### Adding Scale Collection

**Scale** is an array object and you can use the inner tag for it. Object in the array collection (i.e. border) is extended with hyphen in the same tag.

**Example:** e-border-width and e-border-color.

#### HTML

```
<!--To Render the Circular gauge-->
<ej-circulargauge id="CircularGauge1">
  <!--Adding Scale collection to the Circular gauge-->
  <e-scales>
    <e-scale e-showRanges="true" e-startAngle="122" e-sweepAngle="296"
    e-radius="130" e-showScaleBar="true" e-size="1" e-maximum="120"
    e-majorIntervalValue="20" e-minorIntervalValue="10"
    e-border-width="0.5">
  </e-scale>
</e-scales>
</ej-circulargauge>
```

Execute the above code to render the following output.



Value 50

#### Adding Pointer Collection

**Pointer** is an array object and you can use the inner tag for it. Object in the array collection (i.e. pointer cap) is extended with hyphen in the same tag.

**Example:** e-pointerCap-radius.

#### HTML

```
<!--To Render the Circular gauge-->
<ej-CircularGauge id="CircularGauge1">
  <!--Adding Scale collection to the Circular gauge-->
  <e-scales>
    <e-scale>
      <!--Adding pointer collection to the scale collection-->
      <e-pointers>
        <e-pointer e-showBackNeedle="true" e-backNeedleLength="20"
          e-length="95" e-width="7" e-value="80"
          e-pointerCap-radius="12">
        </e-pointer>
      </e-pointers>
    </e-scale>
  </e-scales>
</ej-CircularGauge>
```

Execute the above code to render the output as follows.



Value

### Adding Label Collection

**Label** is also an array object. You can use the inner tag for it.

### HTML

```
<!--To Render the Circular gauge-->
<ej-CircularGauge id="CircularGauge1">
  <!--Adding Scale collection to the Circular gauge-->
  <e-scales>
    <e-scale>
      <!--Adding pointer collection to the scale collection-->
      <e-pointers>...</e-pointers>
      <!--Adding labels collection to the scale collection-->
      <e-labels>
        <e-label e-color="#8c8c8c">
        </e-label>
      </e-labels>
    </e-scale>
  </e-scales>
</ej-CircularGauge>
```

Execute the above code to render the following output.



### Adding Tick Collection

**Tick** is an array object. You can use the inner tag for it.

### HTML

```
<!--To Render the Circular gauge-->
<ej-CircularGauge id="CircularGauge1">
  <!--Adding Scale collection to the Circular gauge-->
  <e-scales>
    <e-scale>
      <!--Adding pointer collection to the scale collection-->
      <e-pointers>...</e-pointers>
      <!--Adding labels collection to the scale collection-->
      <e-labels>...</e-labels>
      <!--Adding ticks collection to the scale collection-->
      <e-ticks>
        <e-tick e-type="major" e-distanceFromScale="2" e-height="16"
        e-width="1" e-color="#8c8c8c">
        </e-tick>
        <e-tick e-type="minor" e-distanceFromScale="2" e-height="8"
        e-width="1" e-color="#8c8c8c">
        </e-tick>
      </e-ticks>
    </e-scale>
  </e-scales>
</ej-CircularGauge>
```

Execute the above code to render the following output.





Value

### Adding Range Collection

**Range** is an array object. You can use the inner tag for it. Object in the array collection (i.e. border) is extended with hyphen in the same tag.

**Example:** e-border-color.

### HTML

```
<!--To Render the Circular gauge-->
<ej-circulargauge id="CircularGauge1">
  <!--Adding Scale collection to the Circular gauge-->
  <e-scales>
    <e-scale>
      <!--Adding pointer collection to the scale collection-->
      <e-pointers>...</e-pointers>
      <!--Adding labels collection to the scale collection-->
      <e-labels>...</e-labels>
      <!--Adding ticks collection to the scale collection-->
      <e-ticks>...</e-ticks>
      <!--Adding ranges collection to the scale collection-->
      <e-ranges>
        <e-range e-distanceFromScale="-30" e-startValue="0" e-endValue="70">
        </e-range>
        <e-range e-distanceFromScale="-30" e-startValue="70"
e-endValue="110" e-backgroundColor="#fc0606"
e-border-color="#fc0606">
        </e-range>
        <e-range e-distanceFromScale="-30" e-startValue="110"
e-endValue="120" e-backgroundColor="#f5b43f"
e-border-color="#f5b43f">
        </e-range>
      </e-ranges>
    </e-scale>
  </e-scales>
</ej-circulargauge>
```

Execute the above code to render the following output.



### Two Way Binding

**Circular Gauge** support the two way binding for the property **value**, **minimum** and **maximum** as mentioned earlier. The following code example explains how to achieve the two way binding in **Circular Gauge**.

### HTML

```
<!doctype html>
<html ng-app="syncApp">
<head>
  <!--Refer the necessary script here-->
</head>
<body ng-controller="CircularGauge">
  <div id="linearframe">
    <ej-circulargauge id="CircularGauge1" e-backgroundcolor="transparent" e-
    value="number" e-width="500" e-readonly="false" e-load="loadGaugeTheme" e-
    enableanimation="false">
      <e-scales>
        <e-scale e-showRanges="true" e-startAngle="122" e-sweepAngle="296"
        e-radius="130" e-showScaleBar="true" e-size="1"
        <!--binding maximum value using angular JS -->
        e-maximum="maximum"
        <!--binding minimum value using angular JS -->
        e-minimum="minimum"
        e-majorIntervalValue="20"
        e-minorIntervalValue="10" e-border-width="0.5">
      <e-pointers>
        <e-pointer e-showBackNeedle="true" e-backNeedleLength="20"
        e-length="95" e-width="7"
        <!--binding pointer value using angular JS -->
        e-value="number"
        e-pointerCap-radius="12">
      </e-pointer>
```

```

</e-pointers>
<e-labels>
<e-label e-color="#8c8c8c"></e-label>
</e-labels>
<e-ticks>
<e-tick e-type="major" e-distanceFromScale="2" e-height="16"
e-width="1" e-color="#8c8c8c"></e-tick>
<e-tick e-type="minor" e-distanceFromScale="2" e-height="8"
e-width="1" e-color="#8c8c8c"></e-tick>
</e-ticks>
<e-ranges>
<e-range e-distanceFromScale="-30" e-startValue="0" e-endValue="70"></e-
range>
<e-range e-distanceFromScale="-30" e-startValue="70"
e-endValue="110" e-backgroundColor="#fc0606"
e-border-color="#fc0606"></e-range>
<e-range e-distanceFromScale="-30" e-startValue="110"
e-endValue="120" e-backgroundColor="#f5b43f"
e-border-color="#f5b43f"></e-range>
</e-ranges>
</e-scale>
</e-scales>
</ej-circulargauge>
</div>
<input type="text" id="txtMax" e-value="number" ej-numerictextbox **ng-
model="number" e-decimalplaces="2" e-showspinbutton="false"
Style="width:110px"/>
<script type="text/javascript">
<!--binding the value to the scope variables in application controller-->
angular.module('syncApp', ['ejangular'])
.controller('CircularGauge', function ($scope) {
$scope.number = 50;
$scope.minimum = 0;
$scope.maximum = 120;
});
</script>
</body>
</html>

```

Execute the above code to render the following output.



## ColorPicker

### Overview

The **ColorPicker** control provides you a rich visual interface for color selection. You can select the color from the professionally designed palettes or custom color. By clicking a point on the color, you can change the active color to the color that is located under the pointer.

You can also choose colors in different specifications; red-green-blue-alpha (opacity) (RGBA), hue-saturation-value (HSV) and hexadecimal (HEX). The **ColorPicker** provides a selection of basic colors, standard presets, custom colors and color swatches.

### Getting Started

Using the following steps, you can create a **TypeScript** ColorPicker component.

#### Creating an ColorPicker in TypeScript

You can create a **TypeScript** application with the help of the given [TypeScript Getting Started Documentation](#).

To create a ColorPicker, add a `input` element with the HTML `id` attribute and pre-defined options set to it.

#### HTML

```
<input id="ColorPicker" type="text" />
<script src="app.js"></script>
```

- Create app.ts file and use the below content

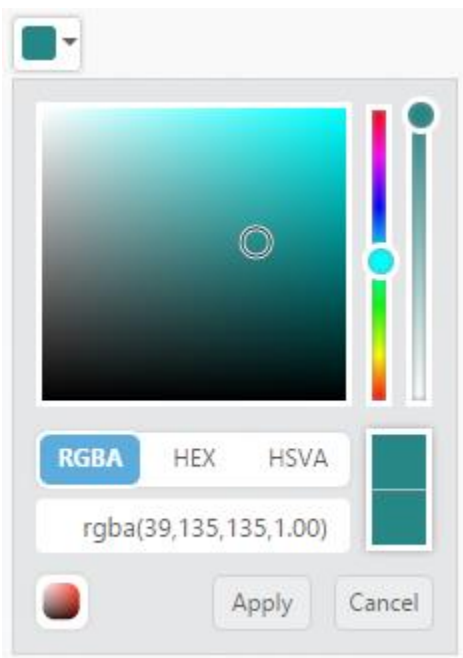
#### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
```

```
module ColorPickerComponent {  
  var colorPickObj = new ej.ColorPicker($("#ColorPicker"), {  
    value: "#278787"  
  })  
}
```

- Now build your application, so that the **app.ts** file will be compiled and automatically generate the **app.js** file which is added to your project (User has nothing to do with this file). Now, whatever code changes that you make in **app.ts** file will be reflected in **app.js** file by compiling and building the application.

Execution of above code will render the following output.



## Behavior Settings

### showPreview

The **ColorPicker** control provides live preview support for current cursor selection color and selected color. **showPreview** property allows you to preview the selected color in the picker or from the palette.

The **showPreview** property is Boolean type and its default value is true.

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget

### HTML

```
<input id="colorpick" type="text" />  
<script src="app.js"></script>
```

- Create **app.ts** file and use the below content

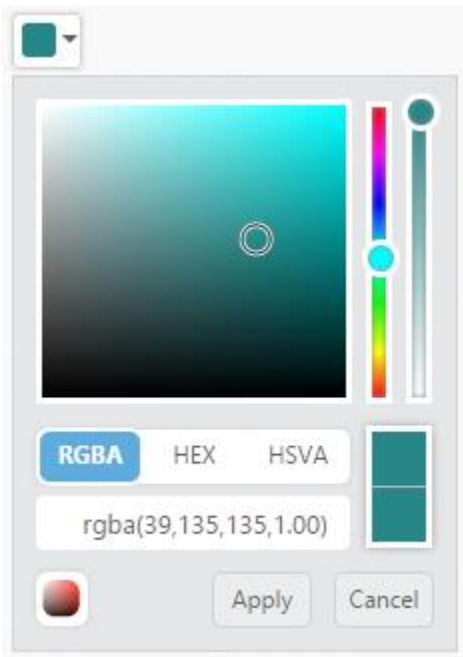
### JS

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ColorPickerComponent {
  var colorPickObj = new ej.ColorPicker($("#colorpick"), {
    value: "#278787",
    showPreview: true
  })
}

```

The following screenshot displays the output of the above code example.



#### showRecentColors

The **ColorPicker** control allows you to store the color values in custom list by using **showRecentColors** property. The **ColorPicker** keeps up to 11 colors in a custom list. By clicking the add button, the selected color from picker or palette gets added in the recent color list.

The **showRecentColors** property is Boolean type and its default value is false.

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget.

#### HTML

```

<input id="colorpick" type="text" />
<script src="app.js"></script>

```

- Create app.ts file and use the below content

#### JS

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ColorPickerComponent {
  var colorPickObj = new ej.ColorPicker($("#colorpick"), {

```

```
value: "#278787",
showRecentColors: true
})
}
```

The following screenshot displays the output of the above code example.



### enableOpacity

The **ColorPicker** control allows you to enable or disable the opacity slider. You can achieve this by using the **enableOpacity** property.

The **enableOpacity** property is Boolean type and its default value is true.

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget

### HTML

```
<input id="colorpick" type="text" />
<script src="app.js"></script>
```

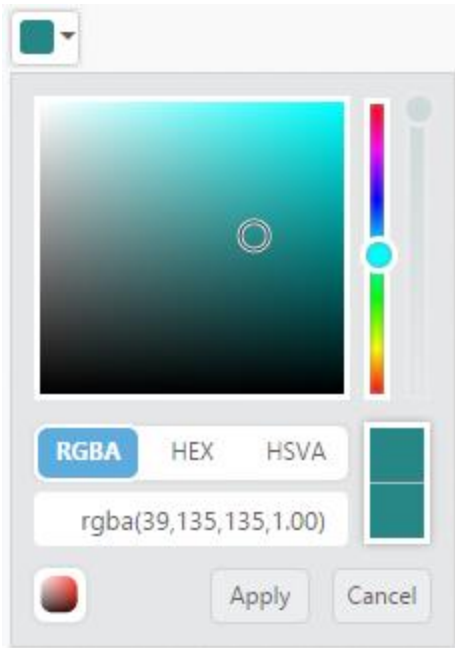
- Create app.ts file and use the below content

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ColorPickerComponent {
var colorPickObj = new ej.ColorPicker($("#colorpick"), {
```

```
value: "#278787",
enableOpacity: false
})
}
```

The following screenshot displays the output of the above code example.



### columns

The palette model consists of color values in the rows and columns order. Palette only consists of predefined colors and allows you to select anyone color from it. The **columns** property allows you to modify the number of columns in palette model.

The **columns** property is Number type and its default value is 10.

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget

### HTML

```
<input id="colorpick" type="text" />
<script src="app.js"></script>
```

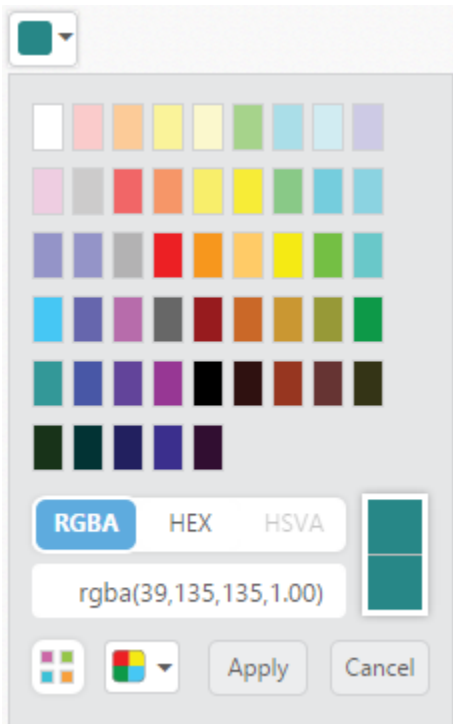
- Create app.ts file and use the below content

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ColorPickerComponent {
var colorPickObj = new ej.ColorPicker($("#colorpick"), {
value: "#278787",
columns: 9
})
}
```



The following screenshot displays the output of the above code example.



## Configure Values

### opacityValue

The **ColorPicker** control allows you to change the opacity value by using the **opacityValue** property. The selected color opacity is adjusted by using the opacityValue.

The **opacityValue** property is Number type and its default value is 10.

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget

### HTML

```
<input type="text" id="colorPicker" />
```

### JAVASCRIPT

```
$(function () {  
  var colorSample = new ej.ColorPicker($("#colorpick"), {  
    value: "#278787",  
    opacityValue: 40  
  });  
});
```

The following screenshot displays the output of the above code example.



button and tooltipText

*buttonText*

The **ColorPicker** control allows you to define the text to be displayed in button elements. You can specify the text by using **buttonText** property. In **ColorPicker** control, popup contains two button elements “Apply” and “Cancel”.

To configure the **buttonText** property for the button elements, use the corresponding default values listed in the following table.

List of Button elements

| Element | Default value |
|---------|---------------|
| apply   | Apply         |
| cancel  | Cancel        |

*tooltipText*

The **ColorPicker** control consists of more number of sub controls and elements. To provide some information about each element and sub control, you can use the tooltip concept and you can achieve this by using **tooltipText** property.

To configure the **tooltipText**, use the following listed elements and its corresponding default value.

List of Tooltip elements

| Element   | Default value |
|-----------|---------------|
| switcher  | Switcher      |
| addbutton | Add Color     |

|               |                |
|---------------|----------------|
| basic         | Basic          |
| monochrome    | Mono Chrome    |
| flatcolors    | Flat Colors    |
| seawolf       | Sea Wolf       |
| webcolors     | Web Colors     |
| sandy         | Sandy          |
| pinkshades    | Pink Shades    |
| misty         | Misty          |
| vintage       | Vintage        |
| moonlight     | Moon Light     |
| candycrush    | Candy Crush    |
| currentcolor  | Current Color  |
| selectedcolor | Selected Color |
| citrus        | Citrus         |

When it is necessary to set the button text and tooltipText values in **Spanish** culture, the **ColorPicker** allow you to define the culture values to **buttonText** and **tooltipText** property. The following section explains on how to define the Spanish culture values to **ColorPicker** control.

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget.

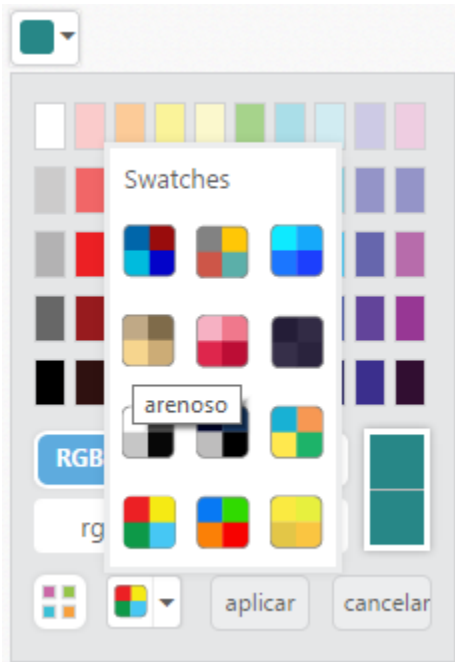
#### HTML

```
<input type="text" id="colorPicker" />
```

#### JAVASCRIPT

```
$(function () {
  var colorSample = new ej.ColorPicker($("#colorpick"), {
    value: "#278787",
    //spanish culture values
    buttonText: { apply: "aplicar ", cancel: "cancelar" }, tooltipText: { sandy:
    "arenoso" }
  });
});
```

The following screenshot displays the output of the above code example.



## Appearance and Styling

### modelType

The **ColorPicker** allows you to define the model type to be displayed in control at initial time by using the property **modelType**.

The **modelType** property is Enum type and its default value is **default**.

List of modelType

| ModelType | Syntax   | Description  |
|-----------|--|--|
| Default   | modelType:<br>ej.ColorPicker.ModelType.Default | Control rendered with both model. You can switch to palette or picker model. |
| Picker    | modelType: ej.ColorPicker.ModelType.Picker     | Control rendered with picker model only.                                     |
| Palette   | modelType:<br>ej.ColorPicker.ModelType.Palette | Control rendered with palette model only.                                    |

In the following code example, the **ColorPicker** popup model type is set as **palette** when you drop down the **ColorPicker** popup.

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget

### HTML

```
<input type="text" id="colorPicker" />
```

### JAVASCRIPT

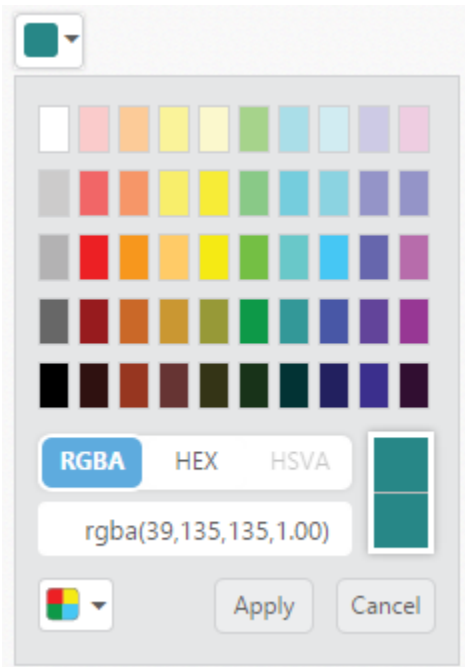
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```

module ColorPickerComponent {
  $(function () {
    var colorSample = new ej.ColorPicker($("#colorpick"), {
      modelType: "palette",
      value: "#278787"
    });
  });
}

```

The following screenshot displays the output of the above code example.



#### palette

The **ColorPicker** allow you to define the palette type to be displayed in control at initial time by using the **palette** property. The **palette** property is Enum type and its default value is **basicpalette**.

List of palette

| Palette       | Syntax   | Description   | Dependent Property                              |
|---------------|--|---|---|
| BasicPalette  | palette:<br>ej.ColorPicker.Palette.BasicPalette  | The palette model rendered with predefined color values.          | modelType :<br>ej.ColorPicker.ModelType.Palette |
| CustomPalette | palette:<br>ej.ColorPicker.Palette.CustomPalette | The palette model renders with the specified custom color values. |   |

### basicpalette

The **basicpalette** type renders with predefined color values. The **basicpalette** model has 12 different preset patterns. Each pattern consists of 50 colors and over 600 colors are available by default.

### presetType

The **ColorPicker** control allows you to define the preset model to be rendered initially in palette type. This can be achieved by using the “presetType” property. Totally 12 types of presets are available.

The **presetType** property is Enum type and its default value is “basic”.

Property Table

| PresetType | Syntax  | Dependent Property                           |
|------------|---|--|
| Basic      | presetType:<br>ej.ColorPicker.PresetType.Basic      | palette: ej.ColorPicker.Palette.BasicPalette |
| MonoChrome | presetType:<br>ej.ColorPicker.PresetType.MonoChrome |  |
| FlatColors | presetType:<br>ej.ColorPicker.PresetType.FlatColors |  |
| SeaWolf    | presetType:<br>ej.ColorPicker.PresetType.SeaWolf    |  |
| WebColors  | presetType:<br>ej.ColorPicker.PresetType.WebColors  |  |
| Sandy      | presetType:<br>ej.ColorPicker.PresetType.Sandy      |  |
| PinkShades | presetType:<br>ej.ColorPicker.PresetType.PinkShades |  |
| Misty      | presetType:<br>ej.ColorPicker.PresetType.Misty      |  |
| Citrus     | presetType:<br>ej.ColorPicker.PresetType.Citrus     |  |
| Vintage    | presetType:<br>ej.ColorPicker.PresetType.Vintage    |  |
| MoonLight  | presetType:<br>ej.ColorPicker.PresetType.MoonLight  |  |
| CandyCrush | presetType:<br>ej.ColorPicker.PresetType.Candycrush |  |

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget

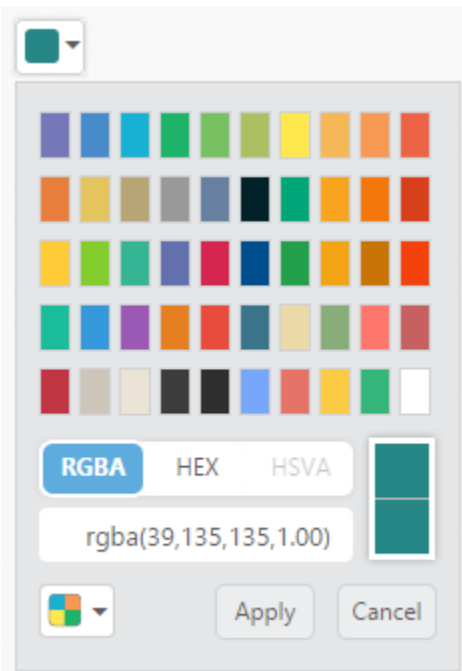
### HTML

```
<input type="text" id="colorPicker" />
```

### JAVASCRIPT

```
$(function () {
  var colorSample = new ej.ColorPicker($("#colorpick"), {
    modelType: "palette",
    value: "#278787",
    presetType: "flatcolors"
  });
});
```

The following screenshot displays the output of the above code example.



### custompalette

The **ColorPicker** control allows you to define the custom colors in the palette model by using **palette** property. Custom palettes are created by passing a comma delimited string of **HEX** values or an array of colors in **custom** property. The custompalette model is only applicable when you set modelType as **palette**.

The **custompalette** property is a dependent property of **palette** and **modelType** property.

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget

### HTML

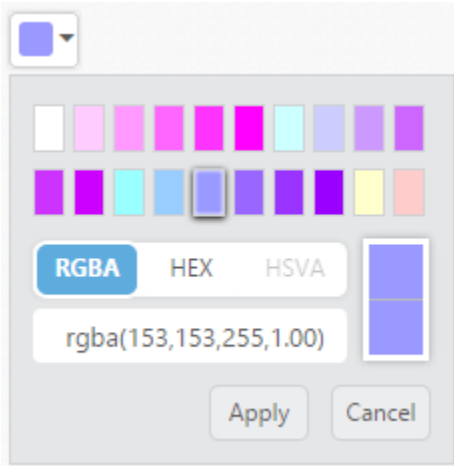
```
<input type="text" id="colorPicker" />
```

### JAVASCRIPT

```
$(function () {
  var colorSample = new ej.ColorPicker($("#colorpick"), {
    value: "#9999ff", palette: "custompalette", modelType: "palette",
  });
});
```

```
custom: ["ffffff", "ffccff", "ff99ff", "ff66ff", "ff33ff", "ff00ff",
"ccffff", "ccccff", "cc99ff", "cc66ff", "cc33ff", "cc00ff", "99ffff",
"99ccff", "9999ff", "9966ff", "9933ff", "9900ff", "ffffcc", "ffcccc"]
});
});
```

The following screenshot displays the output of the above code example.



### displayInline

The **ColorPicker** control allows you to embed the popup in the order of DOM element flow by using the **displayInline** property. Using **displayInline** property to make **ColorPicker** popup always in visible state. Also associate ColorPicker with <div> element instead of input.

The **displayInline** property is Boolean type and its default value is false.

The following steps explain you how to get the **ColorPicker** popup in **DisplayInline** state.

In the **HTML** page, add a <div> element to render **ColorPicker** widget

### HTML

```
<div id="colorPicker" ></div>
```

### JAVASCRIPT

```
$(function () {
var colorSample = new ej.ColorPicker($("#colorpick"), {
value: "#278787",
displayInline: true
});
});
```

The following screenshot displays the output of the above code example.





### Theme Support

The **ColorPicker** control supports rich appearance. It supports 12 different themes of **Essential JavaScript** and bootstrap themes. To use these twelve themes, refer the themes files in HTML page.

You require two style sheets to apply styles to ColorPicker control; one ej.widgets.core.min.css and one ej.theme.min.css. When you use ej.widgets.all.min.css then, it is not necessary to use ej.widgets.core.min.css and ej.theme.min.css because ej.widgets.all.min.css is a combination of these two.

The core style sheet applies styles related to positioning and size, but are not related to the color scheme and always require the control to look correct and function properly. The theme style sheet applies theme-specific styles like colors and backgrounds.

The following list is the twelve themes supported by ColorPicker:

- default-theme
- flat-azure-dark
- flat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget.

### HTML

```
<!doctype html>
<html>
<head>
<title>Essential Studio for JavaScript : ColorPicker - Built-in Theme
Support</title>
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0"
charset="utf-8" />
<link href="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/flat-
lime-dark/ej.web.all.min.css" rel="stylesheet" />
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"> </script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"> </script>
</head>
<body>
<div id="control">
<input id="colorPicker" type="text" />
</div>
<script>
$(function () {
var colorSample = new ej.ColorPicker($("#colorpick"), {
value: "#278787"
});
});
</script>
</body>
</html>

```

**Note:** jQuery.easing external dependency has been removed from version 14.3.0.49 onwards. Kindly include this jQuery.easing dependency for versions lesser than 14.3.0.49 in order to support animation effects.

The following screenshot displays the output of the above code example.



#### CustomCss

The **ColorPicker** control also allows you to customize its appearance using user-defined CSS and custom skin options such as colors and backgrounds. To apply custom themes use the **cssClass** property. **cssClass** property sets the root class for **ColorPicker** theme.

Using this property you can override the existing styles under the theme style sheet. The theme style sheet applies theme-specific styles like colors and backgrounds. In the following example, the value of **cssClass** property is set as **Light-Blue**. **Light-Blue** is added as root class to **ColorPicker** control at the runtime. From this root class you can customize the **ColorPicker** control theme.

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget

#### HTML

```
<input type="text" id="colorPicker" />
```

#### JAVASCRIPT

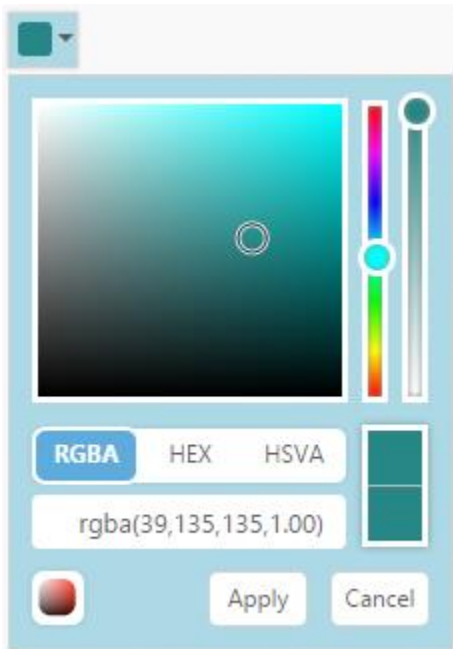
```
$(function () {  
  var colorSample = new ej.ColorPicker($("#colorpick"), {  
    value: "#278787", cssClass: "Light-Blue"  
  });  
});
```

Custom CSS Styles.

#### CSS

```
<style>  
.Light-Blue.e-colorwidget.e-widget, .Light-Blue.e-colorpicker.e-popup,  
.Light-Blue.e-colorwidget .e-in-wrap.e-box .e-select, .Light-Blue.e-  
colorwidget .e-in-wrap.e-box, .Light-Blue.e-colorwidget .e-down-arrow {  
  background: none repeat scroll 0 0 lightblue;  
}  
.Light-Blue.e-colorpicker .e-footer .e-cancelButton.e-btn, .Light-Blue.e-  
colorpicker .e-footer .e-applyButton.e-btn {  
  background: none repeat scroll 0 0 white;  
}  
</style>
```

The following screenshot displays the output of above steps.



### Keyboard Interaction

You can use **keyboard** shortcut keys as an alternative to the mouse while using the **ColorPicker** widget. The **ColorPicker** widget allows you to perform all kinds of actions using **keyboard** shortcuts.

Keyboard shortcut keys

| Shortcut Key | Description                          |
|--------------|--------------------------------------|
| Alt + j      | Focuses into the ColorPicker control |
| Enter        | Open / Close the Popup               |
| Up           | Increase the brightness value        |
| Down         | Decrease the brightness value        |
| Right        | Increase the saturation value        |
| Left         | Decrease the saturation value        |
| Enter        | Choose the current color             |
| Esc          | Closes the popup                     |
| Tab          | Choose the next element              |
| Home         | Downwards to value 0                 |
| End          | Upwards to value 100                 |

### Configure Keyboard Interaction

The following steps explain how you can enable **keyboard** interaction for **ColorPicker** textbox.

In the **HTML** page, add an **<input>** element to configure the **ColorPicker** widget and enable keyboard interaction by the **access key** property.

### HTML

```
<input type="text" id="colorPicker" />
```

### JAVASCRIPT

```
$(function () {  
    var colorSample = new ej.ColorPicker($("#colorpick"), {  
        value: "#278787"  
    });  
    $(document).on("keydown", function (e) {  
        if (e.altKey && e.keyCode === 74) { // j- key code.  
            $("#colorPickerWrapper").focus();  
        }  
    });  
});
```

## Miscellaneous

### getValue

The **getValue()** method in **ColorPicker** returns the hexadecimal value.

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget

### HTML

```
<input type="text" id="colorPicker" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module ColorPickerComponent {  
    $(function () {  
        var ColorObj;  
        var colorSample = new ej.ColorPicker($("#colorpick"), {  
            ColorObj = $("#colorPicker").ejColorPicker({ value: "#278787"  
        }).data('ejColorPicker');  
        ColorObj.getValue();  
    });  
});  
}
```

### setValue

The **setValue()** method in **ColorPicker** is used to set the color value. The given value is in hexadecimal form.

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget

### HTML

```
<input type="text" id="colorPicker" />
```

## JAVASCRIPT

```
$(function () {  
  var colorSample = new ej.ColorPicker($("#colorpick"), {  
  })  
  colorSample.setValue("#278787");  
});
```

## getColor

The **getColor()** method in **ColorPicker** control returns the color value in r,g,b,a form.

In the **HTML** page, add a **<input>** element to render **ColorPicker** widget

## HTML

```
<input type="text" id="colorPicker" />
```

## JAVASCRIPT

```
$(function () {  
  var colorSample = new ej.ColorPicker($("#colorpick"), {  
    value: "#278787"  
  })  
  colorSample.getColor();  
});
```

# CurrencyTextbox

## Overview

**Essential JavaScript CurrencyTextBox** is used to display only currency values. It has Spin buttons to increase or decrease the values in the Text Box.

## Key Features

- **Min and Max Values** — Specifies value range for the CurrencyTextBox.
- **Spin Buttons** — Allows you to increase or decrease the current value in the CurrencyTextBox.
- **Step Value** — Allows you to increment or decrement the current value by step value.
- **Globalization** — Essential JavaScript CurrencyTextBox provides **Globalization** support. These controls use ej.globalize.js file to globalize the number format, and parse numbers according to the culture.
- **Keyboard Navigation** — You can interact with CurrencyTextBox by using keyboard.
- **RTL Support** — Support for right to left alignment of CurrencyTextBox input.
- **Decimal Values** — You can configure CurrencyTextBox to accept decimal values.
- **Themes** — Essential JavaScript CurrencyTextBox consist of 17 built-in themes, and also support custom skins for creating user-defined themes.

## Getting Started

Using the following steps, you can create a **Typescript** CurrencyTextbox component.

### Creating an CurrencyTextbox in TypeScript

You can create a **TypeScript** application with the help of the given [Typescript Getting Started Documentation](#).

Within an index.html file and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>TypeScript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.web
.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add Textbox sample here-->
</body>
</html>
```

The CurrencyTextbox can be created from a `input` element with the HTML `id` attribute and pre-defined options set to it.

#### HTML

```
<input id="currency" type="text" />
<script src="app.js"></script>
```

- Create app.ts file and use the below content

#### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
var cur = new ej.CurrencyTextbox($("#currency"), {
value: 100,
name: "currency",
width: "100%"
});
}
```

- Now build your application, so that the **app.ts** file will be compiled and automatically generated the **app.js** file which is added to your project (User have nothing to do with this file). Now, whatever code changes that you make in **app.ts** file will be reflected in app.js file by compiling build the application.

Execution of above code will render the following output.



set min and max values

- To set the maximum/ending value of the Textbox, you can use the `maxValue` property. Data type of this property is "number".

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var cur = new ej.CurrencyTextbox($("#currency"), {
    value: 100,
    maxValue: 1000,
    name: "currency",
    width: "100%"
  });
}
```

- To set the minimum/starting value of the Textbox, you can use the `minValue` property. Data type of this property is "number".

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module SliderComponent {
  $(function () {
    var number = new ej.NumericTextbox($("#numeric"), {
      value: 30,
      minValue: 1,
      maxValue: 100,
      name: "numeric",
      width: "100%"
    });
  });
}
```

## Behavior Settings

### Decimal Places

The property **decimalPlaces** declares the decimal point to the value of **CurrencyTextBox** control. The default value of `decimalPlaces` is 0 in **CurrencyTextBox** control. To set the `decimalPlaces` to "-1", that allows the decimals without any limit in **CurrencyTextBox** control.

#### Configure Decimal Places

The following steps explain the implementation of `decimalPlaces` in **CurrencyTextBox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **CurrencyTextBox** control.



## HTML

```
<input id="currency" type="text" />
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
  $(function () {
    var number = new ej.CurrencyTextbox($("#currency"), {
      value: 555,
      decimalPlaces: 4
    });
  });
}
```

The output for CurrencyTextbox with decimalPlaces is as follows.



## Persistence Support

The **CurrencyTextbox** widget provides the state maintenance support. You can maintain the previous changes made in the control after a page loads.

### Configure Persistence Support

The following steps explain the implementation of **enablePersistence** in CurrencyTextbox.

In the **HTML** page set the corresponding **<input>** elements for rendering CurrencyTextbox control.

## HTML

```
<input id="currency" type="text" />
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
  $(function () {
    var number = new ej.CurrencyTextbox($("#currency"), {
      value: 33,
      enablePersistence: true
    });
  });
}
```

The output for **CurrencyTextbox** with **enablePersistence** is as follows. You can change the value of CurrencyTextbox and reload the web page.



CurrencyTextbox at initial load



CurrencyTextbox after changing the value and a page load

### Strict Mode Support

The CurrencyTextbox widget allows you to use the strict mode option by setting the **enableStrictMode** property. You can set the **minValue** and **maxValue** to the controls to enable strict mode functionality. When the CurrencyTextbox value exceeds the **maxValue**, it restricts the exceeded value and returns the **maxValue**. Likewise when the CurrencyTextbox value goes below **minValue**, it restricts the new value and returns the **minValue**. When this property is true, it will not restrict the specified value and an error class will be added to indicate wrong value is provided to the CurrencyTextbox.

### Configure Strict Mode Support

The following steps explain the implementation of **enableStrictMode** in CurrencyTextbox .

In the **HTML** page set the corresponding **<input>** elements for rendering **CurrencyTextbox** control.

### HTML

```
<input id="currency" type="text" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
  $(function () {
    var number = new ej.CurrencyTextbox($("#currency"), {
      value: 10, //value(10) exceeds maxValue(8), so it will returns 8.
      minValue: -1,
      maxValue: 8,
      enableStrictMode: true
    });
  });
}
```

The output for **CurrencyTextbox** when **enableStrictMode** is **"true"** is as follows.



### Enabled or Disabled

The CurrencyTextbox control has an option to enable or disable its element. You can set the **enabled** property as **"true"** to enable the CurrencyTextbox control.

Also you can enable/disable the **CurrencyTextBox** by using [enable](#) and [disable](#) methods.

#### *Configure Enabled or Disabled*

The following steps explain the implementation of **enabled** in **CurrencyTextBox**.

In the **HTML** page set the corresponding **<input>** elements for rendering CurrencyTextBox control.

#### **HTML**

```
<input id="currency" type="text" />
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.CurrencyTextbox($("#currency"), {
value: 3,
enabled: false
});
});
}
```

The output for CurrencyTextBox when **enabled** is **"false"** and when **enabled** is **"true"**.



CurrencyTextBox with enabled as false



CurrencyTextBox with enabled as true

#### *Adjusting CurrencyTextBox Size*

The CurrencyTextBox size can be modified by using the **height** and **width** properties. You can customize the size of CurrencyTextBox by using these properties.

#### *Configure Height and Width*

The following steps explain the implementation of **height** and **width** in CurrencyTextBox.

In the **HTML** page set the corresponding **<input>** elements for rendering CurrencyTextBox control.

#### **HTML**

```
<input id="currency" type="text" />
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```

module EditorComponent {
  $(function () {
    var number = new ej.CurrencyTextbox($("#currency"), {
      value: 1,
      width: 100, height: 50
    });
  });
}

```

The output for CurrencyTextBox after setting “height” and “width” is as follows.



#### Increment Step

The **incrementStep** property is used to increase or decrease the amount of value in the CurrencyTextBox control.

#### Configure Increment Step

The following steps explain the implementation of **incrementStep** in **CurrencyTextBox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **CurrencyTextBox** control.

#### HTML

```
<input id="currency" type="text" />
```

#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
  $(function () {
    var number = new ej.CurrencyTextbox($("#currency"), {
      value: 5,
      incrementStep: 4
    });
  });
}

```

The output for CurrencyTextBox with **incrementStep** is as follows.



CurrencyTextBox at initial load



CurrencyTextBox after increasing one step

### Define Name

When you have placed the CurrencyTextBox in a form, the **name** property is used to send the field value at form submission. The default value of the **name** property is null.

### Configure Name

The following steps explain the implementation of **name** in **CurrencyTextBox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **CurrencyTextBox** control.

### HTML

```
<input id="currency" type="text" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
  $(function () {
    var number = new ej.CurrencyTextbox($("#currency"), {
      name: "currency"
    });
  });
}
```

### Define Value

The value of **CurrencyTextBox** can be assigned by using the **value** property. The default value for **value** property is null.

You can get the value of **CurrencyTextBox** by using [getValue](#) method.

### Configure Value

The following steps explain the implementation of **value** in **CurrencyTextBox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **CurrencyTextBox** control.

### HTML

```
<input id="currency" type="text" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
  $(function () {
    var number = new ej.CurrencyTextbox($("#currency"), {
      value: 33
    });
  });
}
```

The output for **CurrencyTextBox** with the **value** property is as follows.



Define **maxValue** and **minValue**

#### *maxValue*

The maximum limit value can be assigned to the CurrencyTextBox by using the **maxValue** property. The default value of **maxValue** property is 1.7976931348623157e+308.

#### *minValue*

The minimum limit value can be assigned to the CurrencyTextBox by using the **minValue** property. The default value of **minValue** property is -1.7976931348623157e+308.

#### *Configure maxValue and minValue*

The following steps explain the implementation of **maxValue** and **minValue** in **CurrencyTextBox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **CurrencyTextBox** control.

#### **HTML**

```
<input id="currency" type="text" />
```

#### **JAVASCRIPT**

```
//CurrencyTextBox with maxValue
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
    $(function () {
        var number = new ej.CurrencyTextbox($("#currency"), {
            maxValue: 4,
            value:8
        });
    });
    //CurrencyTextBox with minValue
    $(function () {
        var number = new ej.CurrencyTextbox($("#currency"), {
            minValue: -3,
            value:-8
        });
    });
}
```

The output for **CurrencyTextBox** with basic properties is as follows.



CurrencyTextBox with **maxValue**



CurrencyTextBox with minValue

#### Read Only Support

The CurrencyTextBox supports read only option. When you enable the **readOnly** property to the control, the value cannot be changed in the CurrencyTextBox. You can set the **readOnly** property as “true” to enable this option.

#### Configure Read Only

The following steps explain the implementation of **readOnly** in **CurrencyTextBox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **CurrencyTextBox** control.

#### HTML

```
<input id="currency" type="text" />
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
  $(function () {
    var number = new ej.CurrencyTextbox($("#currency"), {
      value: 3,
      readOnly: true
    });
  });
}
```

The output for CurrencyTextBox when **readOnly** is “true” is as follows. The CurrencyTextBox values cannot be edited or changed.



#### Appearance

##### Theme

The **CurrencyTextBox** control’s style and appearance can be controlled based on CSS classes. In order to apply styles to the CurrencyTextBox control, you need to refer 2 files namely, **ej.widgets.core.min.css** and **ej.theme.min.css**. If the file **ej.web.all.min.css** is referred, then it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.web.all.min.css** is the combination of these two.

By default, there are 17 themes support available for **CurrencyTextBox** control namely:

- bootstrap
- flat-azure
- flat-azure-dark

- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- high-contrast-01
- high-contrast-02
- material
- office-365

### CSS Class

The **CSS** can be customized by using the **cssClass** in the CurrencyTextbox. You can customize the **CurrencyTextbox** with various **cssClass** properties to appear like your desired control.

### Configure CSS Class

The following steps explain the implementation of **cssClass** in **CurrencyTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **CurrencyTextbox** control.

### HTML

```
<input id="currency" type="text" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
  $(function () {
    var number = new ej.CurrencyTextbox($("#currency"), {
      value: 3,
      cssClass: "customCss"
    });
  });
}
```

Customize the CSS properties in custom CSS class.

### CSS

```
<style>
.customCss .e-box {
  border-color: #9d241b;
}
.customCss .e-input {
  background-color: #f6db8d;
}
.customCss .e-select {
  background-color: #ecf6ac;
```



```
border-color: #3c36e7;
}
</style>
```

The output for CurrencyTextbox after applying **cssClass** is as follows.



### Rounded Corner Support

The CurrencyTextbox provides you with rounded corner support whose appearance is different from normal textbox controls.

#### Configure Rounded Corner Support

The following steps explain the implementation of **showRoundedCorner** in **CurrencyTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **CurrencyTextbox** controls.

#### HTML

```
<input id="currency" type="text" />
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.CurrencyTextbox($("#currency"), {
value: 3,
showRoundedCorner: true
});
});
}
```

The output for CurrencyTextbox when **showRoundedCorner** is **"true"**.



### Spin Button Support

The **CurrencyTextbox** provides you the option as to whether to display the spin button in the widget or remove it from the control by using **showSpinButton** property.

#### Configure Spin Button

The following steps explain the implementation of **showSpinButton** in **CurrencyTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **CurrencyTextbox** control.

#### HTML

```
<input id="currency" type="text" />
```

**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.CurrencyTextbox($("#currency"), {
value: 3,
showSpinButton: true
});
});
}

```

The output for **CurrencyTextbox** when **showSpinButton** is “true”.



CurrencyTextbox with showSpinButton is true



CurrencyTextbox with showSpinButton is false

**Water Mark Text Support**

The **CurrencyTextbox** provide water mark text support. You can display the initial value in the control by water mark.

*Configure Water Mark Text*

The following steps explain the implementation of **watermarkText** in **CurrencyTextbox** .

In the **HTML** page set the corresponding **<input>** elements for rendering **CurrencyTextbox** control.

**HTML**

```

<input id="currency" type="text" />

```

**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.CurrencyTextbox($("#currency"), {
watermarkText: "Currency"
});
});
}

```

The output for CurrencyTextbox after applying **watermarkText** is as follows.



## Globalization Support

**Globalization** is language support based on the culture in CurrencyTextbox. You can achieve the **Globalization** using “**locale**” property in CurrencyTextbox.

The CurrencyTextbox widget provides multi-language support using globalization. You can customize the CurrencyTextbox with your own language style by using this feature. You can change the globalization by using the **locale** property. The default value for **locale** property is **en-US** in CurrencyTextbox controls. Also you can specify the [currencySymbol](#) property when the user wants to overwrite the currency symbol commonly instead of the current culture symbol.

More than 350 culture specific files are available to localize the value. To know more about EJ globalize support, please refer the below link

<https://help.syncfusion.com/js/localization>

**Note:** All the culture-specific script files are available within the below specified location, once you have installed Essential Studio in your machine, therefore it is not necessary to download these files explicitly.

'(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n'

For example, If you have installed the Essential Studio package within C:\Program Files (x86), then navigate to the below location, C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

Refer the below German culture file in head section of HTML page after the reference of **ej.web.all.min.js** file.

### JAVASCRIPT

```
<script src="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/i18n/ej.culture.de-DE.min.js"></script>
```

You can dynamically change the language based on their culture.

### Configure Globalization

The following example describes the way to use localization for **CurrencyTextbox** widgets.

### HTML

```
<input id="currency" type="text" />
```

### JAVASCRIPT

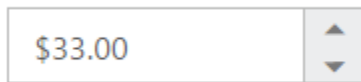
```
// Currency Textbox
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
    $(function) () {
        var number = new ej.CurrencyTextbox($("#currency"), {
```

```
value: 33,
decimalPlaces: 2,
locale: "de-DE"
});
});
}
```

The output for **CurrencyTextBox** with Globalization.



CurrencyTextBox with de-DE locale



CurrencyTextBox with en-US locale

## RTL Support

The **TextBox** provides **RTL** (Right-To-Left) support. The alignment of CurrencyTextBox can be changed from **Left-To-Right** into **Right-To-Left**.

### Enable RTL

The following steps explain the implementation of **enableRTL** in CurrencyTextBox.

In the **HTML** page set the corresponding **<input>** elements for rendering CurrencyTextBox controls.

### HTML

```
<input id="currency" type="text" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var cur = new ej.CurrencyTextbox($("#currency"), {
value: 33,
enableRTL: true
});
});
});
}
```

The output for CurrencyTextBox when **enableRTL** is **"true"** is as follows.



## Keyboard Interaction

With the keyboard navigation enabled in the **CurrencyTextbox** control, it is possible to control the actions of the **CurrencyTextbox** with the provided shortcut keys. Almost all the **CurrencyTextbox** actions that are done through mouse can be controlled with shortcut keys.

The various keyboard shortcuts available within the **CurrencyTextbox** control are discussed in the following table.

### Keyboard Shortcuts

| Shortcut Key          | Description            |
|-----------------------|------------------------|
| <u>Access key</u> + j | Focuses the control    |
| Up                    | Increments the value   |
| Down                  | Decrements the value   |
| Tab                   | Focus the next element |

## Configuring Keyboard Navigation

The following steps explain the implementation of keyboard interaction in **CurrencyTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering CurrencyTextbox controls. Set the **access key** property to the **CurrencyTextbox** control for focusing the control while key is pressed.

### HTML

```
<input id="currency" type="text" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
    $(function () {
        var number = new ej.CurrencyTextbox($("#currency"), {
            value: 33
        });
    });
}
```

Run the above example and press [Access key](#) + j key to focus the **Textbox** widget. Perform provided functionality by using keyboard shortcuts.



## DatePicker

### OverView

The TypeScript DatePicker control provides an intuitive visual interface for selecting a date. Its rich feature set includes functionalities like highlighting special dates, globalization, responsive rendering, accessibility and much more. All properties of DatePicker in TypeScript framework declared with definitions of that properties which will helps us to render the component along with types specified properties.

### Key features

- Formatting the date value.
- Globalization.
- Range the date value to pick.
- Quick picking date by drill down or up.
- State persistence.
- Responsive dimension.
- Flexible customization.
- Custom Styling.
- Built-in jQuery validation.
- Accessibility (keyboard and ARIA).

### Getting Started

This section will explain the Getting started of the DatePicker in TypeScript application, with the step-by-step instructions.

#### Creating an DatePicker in TypeScript

You can create a TypeScript application with the help of the given [Typescript Getting Started Documentation](#).

The DatePicker can be created from a input element with the HTML id attribute and pre-defined options set to it.

#### JAVASCRIPT

```
<input id="datepick" />
```

Create or open (if already exists) app.ts file from TypeScript application, to render the JS component. Then use below code example to render the DatePicker in typescript.

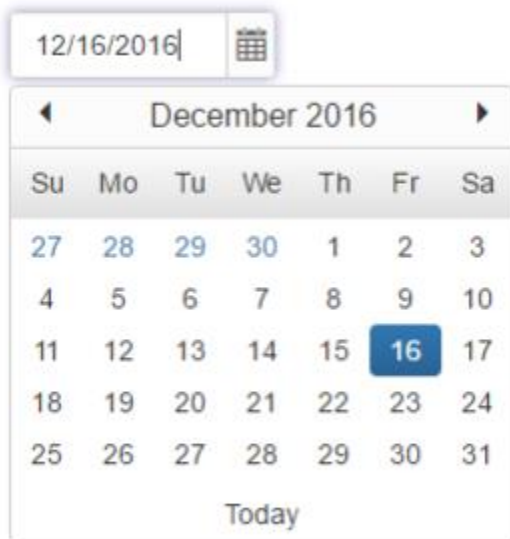
Also, Reference path should be included in the TypeScript definitions of the EJ component

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" >;  
/// <reference path="tsfiles/ej.web.all.d.ts" />;  
module DatePickerComponent {  
  $(function () {  
    var dateSample = new ej.DatePicker($("#datepicker"), {  
      value: new Date()  
    });  
  });  
}
```

Compile and build the application. In compilation time the app.js will be generated which can be referred in index page to render the EJ component.

Execution of above code will render the following output.



#### Configuring the DatePicker properties

All DatePicker APIs can be configured and used in TypeScript like below.

To set the minimum date value of the DatePicker, you can use the minDate property. Data type of this property is "Date/String".

#### JAVASCRIPT

```
module DatePickerComponent {
  $(function () {
    var dateSample = new ej.DatePicker($("#datepicker"), {
      value: new Date(),
      minDate: new Date("11/11/2012"),
    });
  });
}
```

To set the maximum value of the Textbox, you can use the maxDate property. Data type of this property is "Date/String".

#### JAVASCRIPT

```
module DatePickerComponent {
  $(function () {
    var dateSample = new ej.DatePicker($("#datepicker"), {
      value: new Date(),
      maxDate: new Date("11/11/2019"),
    });
  });
}
```

```
});  
}
```

## Behavior Settings

DatePicker has some default behavior settings which helps you to perform more operation by Built-in.

### Selected Date

DatePicker value can be selected through picking date from DatePicker calendar or you can set it by using [value](#) property.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module DatePickerComponent {  
    $(function () {  
        var dateSample = new ej.DatePicker($("#datepicker"), {  
            value: new Date(), // sets the current date  
            change: "onChange" // sets handler to listen value change  
        });  
    });  
    function onChange(args) {  
        //args contains entire model of DatePicker to get the value of all  
        //properties.  
        //alert DatePicker shows the previous date and selected date.  
        alert(" previous date is : " + args.prevDate + " \n selected date is : " +  
            args.value);  
    }  
}
```

DatePicker allows only the valid date to be entered and it should be within the specified range. By default, strict mode is enabled in DatePicker, so it will restrict invalid date and resets to previous date if it's not valid. To know more about strict mode refer [Strict Mode](#).

### Date Range

DatePicker provides an option to restrict the user to pick the date from specified range of value. You can utilize this option by make use of [minDate](#) and [maxDate](#) property.

**minDate** - specifies the minimum date to be picked in DatePicker calendar by disabling below date of minDate.

**maxDate** - specifies the maximum date to be picked in DatePicker calendar by disabling above date of maxDate.

### JAVASCRIPT

```
$(function () {  
    // creates DatePicker from input with current date value  
    var dateSample = new ej.DatePicker($("#datepicker"), {  
        value: new Date(), // sets the current date  
        minDate: new Date("2014/06/03"), // sets min date to be pick able in  
        // DatePicker calendar  
        maxDate: new Date("2014/06/19") // sets max date to be pick able in  
        // DatePicker calendar  
    });  
});
```



```
});
```

**Note:** You can change the 'minDate' and 'maxDate' value dynamically like 'value' property

### Start and Depth Level

Start and depth represents the view of DatePicker calendar. DatePicker calendar has four different level of views, which are month, year, decade and century. It allows you to quick pick date from different months and years by drilling down or up.

By default DatePicker starts with month view and can be drill down into year, decade and century. You can also change the start and depth level view by using [startLevel](#) and [depthLevel](#) property. So that you can initiate DatePicker calendar to view at any level and control the navigation.

- "month" – shows the days in month to pick.
- "year" – shows the months in year to pick.
- "decade" – shows the years in decade to pick.
- "century" – shows the decades in century to pick.

### JAVASCRIPT

```
$(function () {
    // creates DatePicker from input with current date value
    var dateSample = new ej.DatePicker($("#datepicker"), {
        value: new Date(), // sets the current date.
        startLevel: ej.DatePicker.Level.Year, //sets the start view in DatePicker
        calendar.
        depthLevel: ej.DatePicker.Level.Year, //defines when the DatePicker calendar
        to return date.
        dateFormat: "MMMM yyyy" //sets the date format to display in input.
    });
});
```

### Display Inline Mode

DatePicker provides an option to act as calendar by setting the [displayInline](#) property as true. In this mode the DatePicker calendar has been placed open state constantly to pick the date.

You can make use of 'div' element to create DatePicker when you going for inline mode, which gives good visualization as like calendar in page instead of creating with 'input' element.

### HTML

```
<!--div element to create inline mode DatePicker-->
<div id="datepicker"></div>
```

### JAVASCRIPT

```
$(function () {
    // create DatePicker from div
    var dateSample = new ej.DatePicker($("#datepicker"), {
        displayInline: true //sets inline to represent datePicker as DatePicker
        calendar
    });
});
```

### Strict Mode

Strict mode in DatePicker allows you to enter valid or invalid date in input textbox, but an error class will get added to exhibit if it's an invalid date. When you set [enableStrictMode](#) to false, DatePicker allows you to enter only the valid date or else it will resets with previous value.

Also the valid date should be defined in specified range or else it resets to min or maximum date when the entered date is out of range

By default “**enableStrictMode**” is true, so you can enter any value other than date too, but an **error** class (‘. e - error’) will get added to wrapper to highlight the invalid date.

### JAVASCRIPT

```
$(function () {
    // create DatePicker from input
    var dateSample = new ej.DatePicker($("#datepicker"), {
        enableStrictMode: false //sets active strict mode
    });
});
```

You can also override the “**error**” class to highlight when invalid date is entered.

### JAVASCRIPT

```
$(function () {
    // create DatePicker from input
    var dateSample = new ej.DatePicker($("#datepicker"), {
        enableStrictMode: true //sets inactive strict mode
    });
});
```

### CSS

```
<style>
/* error class to highlight invalid date */
.e-error .e-input {
    color: Red; /* highlights invalid date font color in input */
}
</style>
```

### Formatting

Formatting is the way of displaying the date or number as string in some standard format which is based on culture specific or user need.

Below table shows the patterns to format date value.

| Format Pattern | Description   | Result        |
|----------------|---|---------------|
| d              | The day of the month between 1 and 31. Â            | "1" Â to "31" |
| dd             | The day of the month with leading zero if required. | "01" to "31"  |

|      |   |                         |
|------|---|-------------------------|
| ddd  | Abbreviated day name.                               | "Mon" to "Sun"          |
| dddd | The full day name                                   | "Monday" to "Sunday"    |
| M    | The month of the year between 1 - 12                | "1" to "12"             |
| MM   | The month of the year with leading zero if required | "01" to "12"            |
| MMM  | Abbreviated month name                              | "Jan" to "Dec"          |
| MMMM | The full month name                                 | "January" to "December" |
| yy   | The year as a two-digit number                      | "99" or "08"            |
| yyyy | The full four digit year                            | "1999" or "2008"        |

### Date Format

Each culture has some specific date format. Date format defines a format or structure of the displayed date in the textbox. You can change the date format by using [dateFormat](#) property

The standard formats are listed as follows

| Format Name | Formats            |
|-------------|--------------------|
| default     | "M/d/yyyy"         |
| Short       | "d M, y"           |
| Medium      | "d MM, y"          |
| Full        | "dddd, d MMMM, yy" |
| UTC         | "yyyy-MM-dd"       |

By default 'en-US' culture date format is "M/d/yyyy".

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DatePickerComponent {
  $(function () {
    // create DatePicker from input
    var dateSample = new ej.DatePicker($("#datepicker"), {
      value: new Date(), // sets the current date
      locale: "en-US", // sets English -US culture
      dateFormat: "yyyy/dd/MM" // sets the date format to display in input.
    });
  });
}

```

```

});
});
}

```

To get the culture and date format of DatePicker, refer the below code example

#### JAVASCRIPT

```

// create instance for datePicker
// only after control creation we can get dateObj otherwise it throws
exception
var dateObj = $("#datePicker").ejDatePicker('instance');
dateObj.option('locale'); //returns the culture in string
dateObj.option('dateFormat');// returns the date Format in string

```

**Note:** by default date format is based on culture specific. You have to refer the required culture specific files in head section of HTML page in order to localize DatePicker and customize different format for that culture.

#### Header Format

DatePicker calendar consists of header, day header, days and footer section. In which header section shows the current view of DatePicker calendar by displaying the selected day or month or year. It can be formatted as like date format by using [headerFormat](#) property.

#### JAVASCRIPT

```

$(function () {
// creates DatePicker from input
var dateSample = new ej.DatePicker($("#datepicker"), {
headerFormat: "yyyy MMMM" //sets the selected header format to display in
header.
});
});

```

#### Day Header

Day header determines the days name to be displayed in terms of short, medium and long in DatePicker calendar by using [dayHeaderFormat](#) property. Also the DatePicker calendar size varies with this specified values.

#### JAVASCRIPT

```

$(function () {
// create DatePicker from input
var dateSample = new ej.DatePicker($("#datepicker"), {
dayHeaderFormat: ej.DatePicker.Header.Long //sets the day header as long
});
});

```

#### Tooltip with Formatting

DatePicker calendar shows tooltip on hovering the date by specifying the formatted date of hovered date. Its helps you to get clear view about the date going to select. You can show or hide this tooltip option by using [showTooltip](#) property.

You can also change the format of tooltip by using “**tooltipFormat**” property. Below codes example allows to show tooltip and format its value.

### JAVASCRIPT

```
$(function () {
  // creates DatePicker from input
  var dateSample = new ej.DatePicker($("#datepicker"), {
    showTooltip: true, //show tooltip on hovering date on DatePicker calendar
    tooltipFormat: "dd/MM/yy" // sets tooltip for dates in DatePicker calendar
  });
});
```

## Globalization

DatePicker has been provided with Built-in localization support, so that it will be able adapt based on culture specific locale defined for it.

More than 350 culture specific files are available to localize the date. To know more about EJ globalize support, please refer the below link

<https://help.syncfusion.com/js/localization>

**Note:** Seven culture-specific script files are available in the below specified location. For all other culture files, please download from the [GitHub](#) location.

```
(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n
For example, If you have installed the Essential Studio package within C:\Program Files (x86), then
navigate to the below location, C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n
```

To translate our control content from default English to any of the culture, say For example - German language, then you need to refer the ej.culture.de-DE.min.js file in your application,

The **en-US** locale is currently being used as default culture in DatePicker. You can set any other culture to DatePicker by using **Locale** property. Below code example shows German cultured DatePicker.

Refer the below German culture file in head section of HTML page after the reference of **ej.web.all.min.js** file.

### JAVASCRIPT

```
<script src="https://cdn.syncfusion.com/js/assets/i18n/ej.culture.de-DE.min.js"></script>
```

Set German culture to DatePicker at initialization.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DatePickerComponent {
  $(function () {
    // create DatePicker from input with current date value
    var dateSample = new ej.DatePicker($("#datepicker"), {
```

```
value: new Date(), // sets the current date
locale: "de-DE" // sets German culture
});
});
}
```

### Watermark and Today Button Text

By default DatePicker input has “**select date**” as watermark text, you can also change this value by using [watermarkText](#) property. Also there’s a today button in DatePicker calendar which allows you to quick select the current date and its value can be changed by using ‘**buttonText**’ property.

#### JAVASCRIPT

```
$(function () {
    // creates DatePicker from input
    var dateSample = new ej.DatePicker($("#datepicker"), {
        watermarkText: "enter date value", // sets watermark text in input
        buttonText: "current date" // sets button text in DatePicker calendar
    });
});
```

Based on culture specific, only date gets localized but by changing the watermark and button text, you can completely localize the DatePicker UI too.

Refer below code example to update those value based some culture say for example English and German.

#### JAVASCRIPT

```
ej.DatePicker.Locale['en-US'] = {
    watermarkText: "Select date",
    buttonText: 'Today'
};
ej.DatePicker.Locale['de-DE'] = {
    watermarkText: "Zeitraum auswählen",
    buttonText: 'Heute'
};
```

### Accessibility

#### Keyboard Interaction

DatePicker support all possible keyboard interaction, you can enable this option by specifying “**allowKeyboardNavigation**” as true.

You can use the keyboard shortcut keys as an alternative of mouse to interact with DatePicker widget.

Keyboard shortcut keys are,

| Key        | Description                   |
|------------|-------------------------------|
| Alt + Down | Opens the DatePicker calendar |
| Left       | Moves to previous date        |

|              |                             |
|--------------|-----------------------------|
| Right        | Moves to next date          |
| Up           | Moves one week upward       |
| Down         | Moves one week downward     |
| Enter        | Selects the focused date    |
| Esc          | Closes the popup            |
| Ctrl + Up    | Navigates to top view       |
| Ctrl + Down  | Navigates to lower view     |
| Ctrl + Left  | Navigates to previous month |
| Ctrl + Right | Navigates to next month     |

## Customization

DatePicker provides more flexible way to customize the below listed properties.

### Set Dimension

By default DatePicker has standard height and width (height: '30px' and width: '143px'). You can change this height and width by using [height](#) and [width](#) property respectively.

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DatePickerComponent {
  $(function () {
    // creates DatePicker from input
    var dateSample = new ej.DatePicker($("#datepicker"), {
      height: '50px', // sets height as 50 pixel
      width: '300px' // sets width as 300 pixel
    });
  });
}

```

Mostly dimension plays a vital role in web application to get responsive layout in all devices. DatePicker is a form control and you can make it as responsive by specifying its width as "100%".

### JAVASCRIPT

```

$(function () {
  // creates DatePicker from input
  var dateSample = new ej.DatePicker($("#datepicker"), {
    width: '100%' // sets width as 100 percentage
  });
});

```

### Show Footer

You can show or hide the footer of DatePicker calendar by using [showFooter](#) property.

#### JAVASCRIPT

```
$(function () {  
    // creates DatePicker from input  
    var dateSample = new ej.DatePicker($("#datepicker"), {  
        showFooter: false // hides the footer in DatePicker calendar  
    });  
});
```

**Note:** Footer contains the today button to quickly navigate to the current date. By turning off it, you have to select current date by manually.

### Show Popup Button

DatePicker textbox contains a button to open the DatePicker calendar. You can hide this DatePicker calendar button using [showPopupButton](#) value as false.

By hiding this button, you can open the DatePicker by focusing the input textbox element itself.

#### JAVASCRIPT

```
$(function () {  
    // creates DatePicker from input  
    var dateSample = new ej.DatePicker($("#datepicker"), {  
        showPopupButton: false // hides the DatePicker calendar button  
    });  
});
```

### Show Other Months

You can show or hide the other month days from DatePicker calendar by using [showOtherMonths](#) property.

#### JAVASCRIPT

```
$(function () {  
    // creates DatePicker from input  
    var dateSample = new ej.DatePicker($("#datepicker"), {  
        showOtherMonths: false // hides the days of other months in DatePicker calendar  
    });  
});
```

### Highlight Special Date

DatePicker allows you to highlight the special dates in DatePicker calendar by using [specialDates](#) property. It receives the array of JSON data, in which the data should contain the date value, icon CSS class and tooltip as field values with any names.

#### JAVASCRIPT

```
var data = [{ specialdate: "28/06/2015", customClass: "birthday", tooltip: "xxx birthday" }]
```



And you can map those above field names to DatePicker by using [fields](#) property, which holds following members.

| Members   | Description                                       |
|-----------|---|
| date      | It specifies the mapping field of special date    |
| iconClass | It specifies the mapping field of icon CSS class. |
| tooltip   | It specifies the mapping field of tool tip text.  |

### JAVASCRIPT

```
// assign special date values in local variable data
var data = [{ specialdate: "28/06/2015", customClass: "birthday", tooltip:
"xxx birthday" }]
$(function () {
// creates DatePicker from input
var dateSample = new ej.DatePicker($("#datepicker"), {
specialDates: data, // date values in data variable assigned to specialDates
fields: { date: "specialDate", iconCSS: "customClass", tooltip: "toolTip" }
// mapping special date fields
});
});
```

### Validation

DatePicker is a form control and therefore its value to be validated before processing. Client side validation provides a better user experience by responding quickly at browser level. Most common client side validation plugin is [jQuery Validation](#).

DatePicker supports this validation by built-in, so you can do client validation with simple steps to validate and respond validation message. The jQuery validate plugin rules and custom methods can be used in the DatePicker control with the help of two properties, [validationRules](#) and [validationMessage](#).

Before using those properties, you need to add the jQuery validate plugin to your HTML page.

Refer the below jQuery validation plugin script after jQuery script reference.

### HTML

```
<script
src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.validate.m
in.js"></script>
```

### HTML

```
<!--form to submit-->
<form>
<!--input element to create DatePicker-->
<input id="datepicker" />
<!-- submit button -->
```

```
<input type="submit" value="submit" />
</form>
```

jQuery validation library contains some default settings for validating the form. By default, it ignores hidden elements in from validation. If validation gets fail, in-built “error” class will be added to corresponding element. Here you can specify a custom class with your own style using “errorClass” API and you can place the error message in necessary position using “errorPlacement” API. Refer following code block to customize the default jQuery validation settings.

#### JAVASCRIPT

```
$.validator.setDefaults({
  //if we don't set custom class, default "error" class will be added
  errorClass: 'e-validation-error',
  //it specifies the error message display position
  errorPlacement: function (error, element) {
    $(error).insertAfter(element.closest(".e-widget"));
  }
});
```

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module DatePickerComponent {
  $(function () {
    // creates DatePicker from input
    $("#datePicker").ejDatePicker({
      validationRules: { required: true }, // sets the field to be required
      validationMessage: { required: "Required Date value" } // sets validation
      message
    });
  });
}
```

**Note:** jQuery validation works only within the form element.

#### State Persistence

You can sustain the entire widget model of DatePicker even after form post or browser refresh by using [enablePersistence](#) property. So the entire model values such as

- Selected date
- Highlighted date
- Start and depth level

are stored in local storage / cookies of browser before page refreshes and reinitialized with the restored model after refresh.

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module DatePickerComponent {
```

```
$(function () {  
    // creates DatePicker from input  
    var dateSample = new ej.DatePicker($("#datepicker"), {  
        enablePersistence: true // persists the DatePicker model  
    });  
});  
});  
}
```

## Miscellaneous

### Start Day

By default DatePicker calendar starts with “Sunday” and ends with “Monday”. You can redefine this start day by using [startDay](#) property.

Refer below code to start Wednesday as start day.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module DatePickerComponent {  
    $(function () {  
        // creates DatePicker from input  
        var dateSample = new ej.DatePicker($("#datepicker"), {  
            startDay: 3 // sets start day as Wednesday in DatePicker calendar  
        });  
    });  
};  
}
```

### Step Months

DatePicker calendar allows you to quick navigate back and forth from one month to previous or next month by clicking the arrow button. By default it's navigate one by one month. You can also navigate by skipping months in odd or even or any count by using [stepMonths](#) property.

#### JAVASCRIPT

```
$(function () {  
    // creates DatePicker from input  
    var dateSample = new ej.DatePicker($("#datepicker"), {  
        stepMonths: 2 // skips the one months from current month(July to Sept to Nov)  
    });  
});  
});  
}
```

### Read Only

You can make DatePicker as read only by setting [readOnly](#) property as true. It allows only to read the value and it can't be changed by interaction.

#### JAVASCRIPT

```
$(function () {  
    // creates DatePicker from input  
    var dateSample = new ej.DatePicker($("#datepicker"), {  
        readOnly: true //sets DatePicker as read only  
    });  
});  
});  
}
```

```
});
```

### Enable or Disable

You can enable or disable the DatePicker textbox by using [enabled](#) property. In inline mode DatePicker calendar also gets enabled or disabled.

#### JAVASCRIPT

```
$(function () {  
    // creates DatePicker from input  
    var dateSample = new ej.DatePicker($("#datepicker"), {  
        enabled: false //disables the DatePicker textbox  
    });  
});
```

### How to?

Customizing template with range selection between two DatePicker.

You can customize the date field to emphasize the particular dates in DatePicker calendar with help of [specialDates](#) and set the date range using [minDate](#) and [maxDate](#) property. Refer the sample from the link [Hotel Booking](#) to know how to customize date with date range.

### Localize DatePicker with browser specific culture

You can get the browser culture name at page load or document ready state. Based on the culture name, DatePicker can be initiated with that specific culture value by assigning to [locale](#) property. Refer the sample from the link [Browser Specific Culture](#) to create DatePicker with browser specific culture.

### Disable specific dates to restrict user

DatePicker allows you to restrict date selection in specific range by using date range option. But you can also restrict selective date in DatePicker calendar by utilizing [beforeDateCreate](#) event. This event will get triggered at each date creation. So you can disable the selective date in this event to restrict the user.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module DatePickerComponent {  
    $(function () {  
        // creates DatePicker from input  
        $("#datepicker").ejDatePicker({  
            beforeDateCreate: "restrictDate" // handler to listen each date create.  
        });  
    });  
}  
  
//event get triggered for each date create.  
function restrictDate(args) {  
    //date to disable in DatePicker calendar to restrict selection.  
    var disableDate = new Date("09/22/2015");  
    //compares two and adds the disable class.  
    if (+args.date === +disableDate)  
        args.element.addClass('e-disable');  
    // args contains current date and its element.  
}
```

### How to integrate with bootstrap grid system?

DatePicker is responsive control, you have to just set the input element width as 100%. In Bootstrap grid layout use the below code example to get responsive textbox.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DatePickerComponent {
  $(function () {
    // creates DatePicker from input
    var dateSample = new ej.DatePicker($("#datepicker"), {
      width: '100%' // sets width as 100 percentage
    });
  });
}
```

### How to use DatePicker in AngularJS?

You can use DatePicker control in AngularJS Framework. To know more about JS component with AngularJS refer here [JS component in AngularJS](#).

## DateRangePicker

### Overview

The DateRangePicker control displaying two calendars in a web page and allow to pick the two date to form the ranges. Also its calendar consist of flexible option to navigate back and forth from month and years and pick the start and end date in any month. Flexible options of this DateRangePicker provides the easiest way to get the details within the date ranges in application

### Key Features

1. Formatting the date value and time value.
2. Globalization.
3. Preset Ranges
4. Time Picker option
5. Quick picking date by drill down or up.
6. State persistence.
7. Responsive dimension.
8. Flexible customization.
9. Custom Styling

### Getting Started

This section explains you how to render and configure DateRangePicker component in a TypeScript application.

#### Create your first DateRangePicker

1. Create a TypeScript application and refer the dependent modules, script and CSS with the help of given Getting started document.
2. In the index.HTML file, add the input element for rendering DateRangePicker component as given below.

## HTML

```
<input id="daterangepicker" />
```

3. Create a TypeScript file named "app.ts" file and refer the required definition files as given below.

## JS

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

4. Now, initialize the DateRangePicker component by using ej.DateRangePicker method.

## JS

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
$(function () {  
  var sample = new ej.DateRangePicker($("#daterangepicker"));  
});
```

### Run the application

To run the application, navigate the project folder and open command prompt window and execute the following command.

## JS

```
tsc
```

This command compiles the app.ts file to generate a JS file named app.js file.

Refer the app.js file in index.html and browse the html file.

### Get/Set Value

DateRangePicker provides an options to configure all its properties and to get its value. DateRangePicker value can be assigned during initialization or at run time. Below code shows how to assign the values at initialization.

## JS

```
/// initialize DateRangePicker component with Value API  
/// <reference path="../../tsfiles/jquery.d.ts" />  
/// <reference path="../../tsfiles/ej.web.all.d.ts" />  
module DateRangePickerComponent {  
  $(function () {  
    var sample = new ej.DateRangePicker($("#daterangepicker"), {  
      value: "11/1/2013 - 12/3/2019", // sets the date range  
    });  
  });  
}
```

## Behavior Settings

DateRangePicker has some default behavior settings which helps you to perform more operation by Built-in.

### Selected Date Range

#### Value

DateRangePicker value can be selected through picking two date values from available two DatePicker calendar or you can set it by using **value** property.

#### HTML

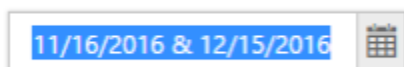
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateRangePickerComponent {
$(function () {
var datetimesample = new ej.DateRangePicker($("#daterangepick"), {
value: "11/1/2016 - 11/2/2017", // sets the date range
onChange: "onChange" // sets handler to listen value change
});
});
}
function onChange(args) {
//args contains entire model of DateRangePicker to get the value of all
properties.
//alert DateRangePicker shows the start date and end date.
alert(" start date is : " + args.startDate + " \n end date is : " +
args.endDate);
}
```

#### Separator

The value of the DateRangePicker popup will presented with two date strings which is separated by **separator** (e.g. ""11/1/2016 - 11/2/2017"). Separator will be "-" by default and this can be changed using API called **separator**. Please check with below code example to setting/changing the separator using **separator** API.

#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateRangePickerComponent {
$(function () {
var datetimesample = new ej.DateRangePicker($("#daterangepick"), {
value: "11/1/2016 - 11/2/2017", // sets the date range
separator: "&" // sets the separator to & instead of "-"
});
});
}
```



### Start Date

Start Date of the date range can be pick from date range picker calendar. We can select the start date from any one of calendar in a popup.

While selecting the date on calendar in following cases, the clicked date will be considered as StartDate.

1. Click on empty calendar (with no start date, range, end date)
2. Click on date which is lesser than existing start date
3. Click on calendar when already there is start date and end date is updated.

Start Date of range, can be set using API called **startDate** also please refer the below code example:

#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateRangePickerComponent {
$(function () {
var datetimeSample = new ej.DateRangePicker($("#daterangepick"), {
startDate: new Date("11/2/2016")
});
});
}
```

**startDate** can be set to popup, by entering the date value into first input box in popup also.

### End Date

End Date of the date range can be selected from popup directly. Else this can be also updated by using the API called "**endDate**".

The Selection next to the **startDate** will be considered as end Date. This selected date should be higher or equal date than the existing start date. Else this selection will be considered as **startDate** as we discussed in **startDate** section above.

Below code will explain to use the **endDate** API to set the end Date in popup.

#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateRangePickerComponent {
$(function () {
var datetimeSample = new ej.DateRangePicker($("#daterangepick"), {
startDate: new Date("11/2/2016"),
endDate: new Date("11/3/2018")
});
});
}
```

End Date can be set to popup by entering the date into the second input box in popup

### Preset Ranges

We can make use of preset range for easy selection on popup. The ranges provided with this API **presetRanges**, will be processed and corresponding label will be added to popup in right side with given



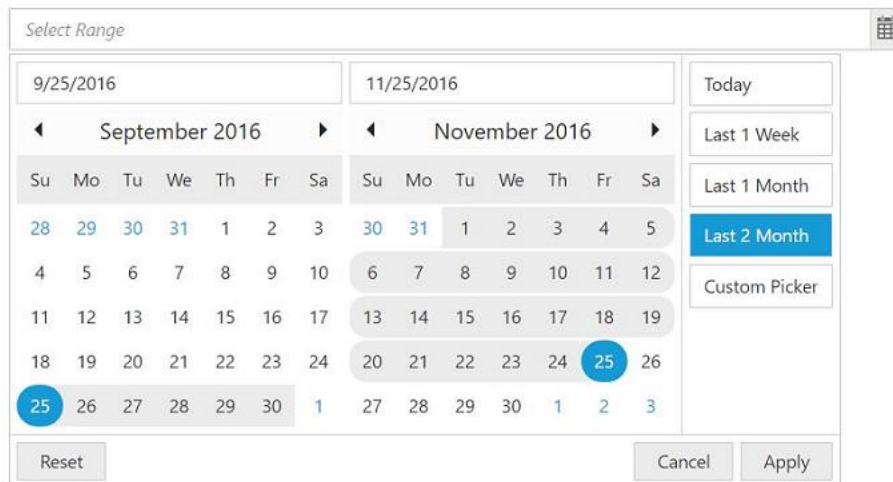
label name. By clicking on these labels the associated ranges will be updated in popup. The below code will explain to use **presetRanges** API.

### HTML

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateRangePickerComponent {
$(function () {
var datetimeSample = new ej.DateRangePicker($("#daterangepicker"), {
ranges: [
{ label: "Today", range: [new Date(), new Date()] },
{ label: "Last 1 Week", range: [new Date(new Date().setDate(new
Date().getDate() - 7)), new Date()] },
{ label: "Last 1 Month", range: [new Date(new Date().setMonth(new
Date().getMonth() - 1)), new Date()] },
{ label: "Last 2 Month", range: [new Date(new Date().setMonth(new
Date().getMonth() - 2)), new Date()] },
],
});
});
}

```



These ranges can be processed and updated to popup by using the **setRange** method also like below code example.

//bind below onClick action to button

### JS

```

function onClick() {
//create instance for dateRangePicker.
// create instance only after control creation, to get dateRangeObj
otherwise it throws exception.
var dateRangeObj = $("#dateRangePicker").ejDateRangePicker('instance');
//call setRange using label
dateRangeObj.setRange("Last 1 Week");
//get value using date range object and displays in alert box
alert(dateRangeObj.option('value'));
}

```

### TimePicker Option

The ranges can be set with time value also by enable the TimePicker in popup using **enableTimePicker** API. Each start date and end date, have separate Time Pickers. Please check with the below code example to enable the time picker.

### JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateRangePickerComponent {
$(function () {
var datetimeSample = new ej.DateRangePicker($("#daterangepicker"), {
enableTimePicker: true
});
});
}

```

The screenshot displays the DateRangePicker control. At the top, a text box shows the selected range: "11/15/2016 12:00 AM - 12/24/2016 12:00 AM" with a calendar icon. Below this, two calendar popups are shown side-by-side. The left calendar is for November 2016, and the right is for December 2016. In the November calendar, the 15th is selected. In the December calendar, the 24th is selected. Both date boxes above the calendars show the selected date and time: "11/15/2016 12:00 AM" and "12/24/2016 12:00 AM". At the bottom of the popups are buttons for "Reset", "Cancel", and "Apply".

### Display Format

Text representation of date and time in their corresponding text box in a control, can be changed using the available API called **dateFormat**, **timeFormat**. By default those values will be set based on culture.

To know about standard patterns for date format. Please check with the format section available in DatePicker [display format section](#).

Also like formatting the date value, the time value also can be formatted. Make use of below details, to create your custom time format using **timeFormat** API.

- hh – Hour.
- mm – Minutes.
- ss – Seconds.
- tt - The AM/PM designator.

### Globalization

DateRangePicker, has built in support with Localization so that you can make use of culture specified DateRangePicker component with your application using **locale** API,

More than 350 culture specific files are available to localize the date. To know more about EJ globalize support, please refer the below link

<https://help.syncfusion.com/js/localization>

Also we have added Localized text files to DateRangePicker for 28 standard culture in below GitHub location,

<https://github.com/syncfusion/ej-global>

### Setting different culture to control

By setting another culture other than English we need to set using **locale** API like below code example.

Also it is necessary to add the corresponding culture file with application and it should be referred after the ej.web.all.min.js file reference.

```
<script src="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/i18n/ej.culture.de-DE.min.js"></script>
```

To get the localized text, it is necessary to add localized text file, else you can also set the localized text as your own like below code example. Adding Localized text, will make completely localized UI to DateRangePicker.

### JS

```
ej.DateRangePicker.Locale['fr-FR'] = {
  ButtonText: {
    apply: "Appliquer",
    cancel: "Annuler",
    reset: "Réinitialiser",
  },
  watermarkText: "Sélectionnez l'intervalle de date",
  customPicker: "sélecteur personnalisé"
};
ej.DateRangePicker.Locale['de-DE'] = {
  ButtonText: {
    apply: "anwenden",
    cancel: "stornieren",
    reset: "zurückstellen",
  },
  watermarkText: "Datumbereich selecteren",
  customPicker: "benutzerdefinierte Picker"
};
```

### Customization

With available flexible options customization of DateRangePicker will be easier.

### Setting Dimension

**Height** and **width** of the DateRangePicker can be changed using corresponding API (**height,width**) like below code examples.

#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateRangePickerComponent {
$(function () {
var datetimeSample = new ej.DateRangePicker($("#daterangepick"), {
height: 50,
width: 300
});
});
}
```

### Rounded Corners

You can make use of **showRoundedCorner** property in order to add rounded borders to the input and popup elements. By default, in DateRangePicker this will be disabled state we can enable this by setting true to this API.

// creates DateRangePicker and setting rounded corner dynamically

#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateRangePickerComponent {
$(function () {
var datetimeSample = new ej.DateRangePicker($("#daterangepick"), {
showRoundedCorner: true
});
});
}
```

### State Persistence

The value of DateRangePicker can be sustained even after form post back and page refreshing by enabling the **enablePersistence** API like below code example.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateRangePickerComponent {
$(function () {
var datetimeSample = new ej.DateRangePicker($("#daterangepick"), {
enablePersistence: true
});
});
}
```

The DateRangePicker Model value will be stored in local storage / cookies of browser before page refreshes and reinitialized with the restored model after refresh.

## Miscellaneous

### [Enable / Disable](#)

The control can be enabled / disabled using public methods, **enable** or **disable**.

### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateRangePickerComponent {
$(function () {
var datetimeSample = new ej.DateRangePicker($("#daterangepick"),
{"disable"});
});
}
```

### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateRangePickerComponent {
$(function () {
var datetimeSample = new ej.DateRangePicker($("#daterangepick"),
{"enable"});
});
}
```

### [Allow Editing](#)

Editing in input box can be disabled by setting the false value to **allowEdit** API. By default this will be false and we can edit the value in input box

### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateRangePickerComponent {
$(function () {
var datetimeSample = new ej.DateRangePicker($("#daterangepick"), {
allowEdit: false
});
});
}
```

### [Clear ranges](#)

To clear the all selection and ranges in a popup we can make use of **clearRanges** method.

### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateRangePickerComponent {
$(function () {
var datetimeSample = new ej.DateRangePicker($("#daterangepick"),
{"clearRanges"});
});
}
```

## DateTimePicker

### Overview

DateTimePicker control is used to input the date and time with a specific format. The DateTimePicker control combines the DatePicker and TimePicker controls so that you can select the date and time with your desired format.

### Key Features

- **DateTime format:** Supports all valid date and time formats.
- **Localization:** Supports localization of different cultures.
- **Persist:** Supports state maintenance during page refresh.
- **RTL:** Supports Right to Left alignment of content in the DateTimePicker control.
- **Themes:** Essential JavaScript controls consist of 17 built-in themes (6 – flat and 6 – gradient effects), and also support custom skin options to set user-defined themes.

---

**Note:** As the DateTimePicker inherits the functionalities of the DatePicker and TimePicker controls, it supports the basic functionalities of both the controls.

---

### Getting Started

This section discloses the details on how to render and configure a DateTimePicker component in a TypeScript application.

#### Create your first DateTimePicker

1. Create a TypeScript application and refer the dependent modules, script and CSS with the help of given getting started document.
2. In the index.HTML file, add the input element for rendering DateTimePicker component as given below.

#### HTML

```
<input id="datetimepick" />
```

3. Create a TypeScript file named "app.ts" file and refer the required definition files as given below.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

4. Now, initialize the DateTimePicker component by using ej.DateTimePicker method.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```
$(function () {
    var sample = new ej.DateTimePicker($("#datetimepick"));
});
```

#### Run the application

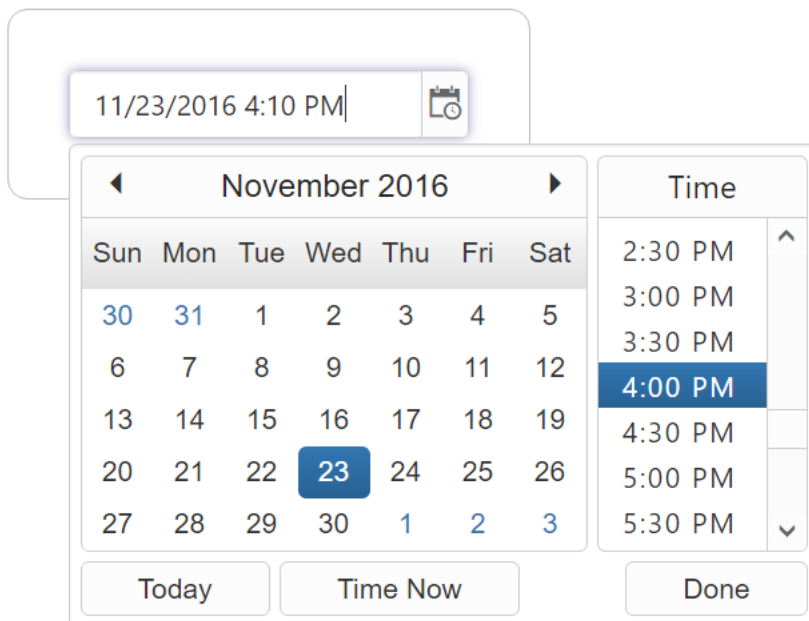
To run the application, navigate the project folder and open command prompt window and execute the following command.

#### JS

```
tsc
```

This command compiles the app.ts file to generate a JS file named app.js file.

Refer the app.js file in index.html and browse the HTML file to see the following output.



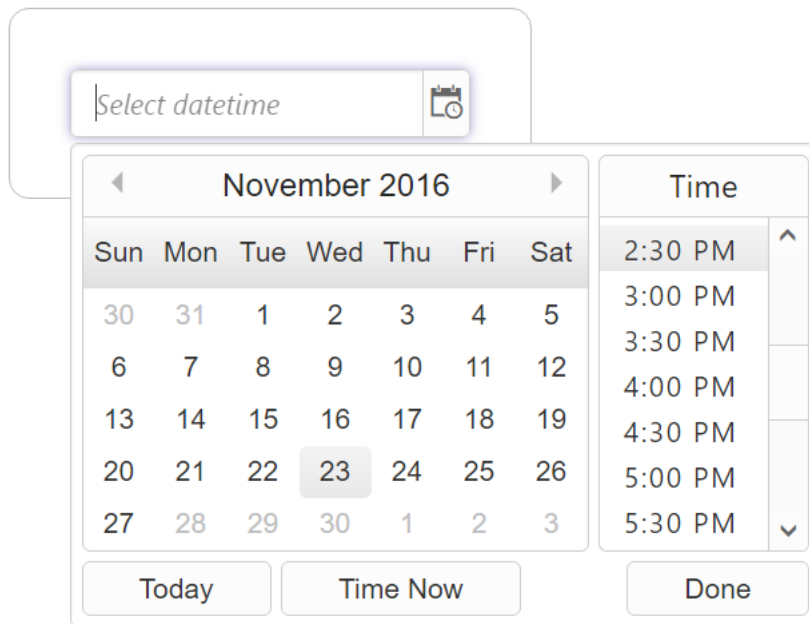
#### Configuring DateTimePicker

EJ DateTimePicker provides API through which you can set the maximum and minimum allowed date and time values. Min and Max date and time values can be set at initialization as show below.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateTimePickerComponent {
    $(function () {
        var sample = new ej.DateTimePicker($("#datetimepick"), { minDateTime: new
        Date("11/1/2016 10:00 AM"), maxDateTime: new Date("11/27/2016 10:00 PM") });
    });
}
```

The following screenshot illustrates the output of above code.



## Display format

### DateTime format

**DateTimePicker** allows you to define the text representation of a date and time value to be displayed in the **DateTimePicker** control. The format specified is achieved by the **dateTimeFormat** property. Default value of this property is **M/d/yyyy h: mm tt**. To change the "Time Popup" display format, "timeDisplayFormat" is used here. The default value of this property is "**h:mm t**".

If your company's website is going to be used all over the world, following the **UTC** time is better. Main benefit of UTC Time is that the time is always guaranteed to be *consistent*. In other words, whenever the time zone of customer is changed you don't have to go back or forth in time from the logging time of the customer to your time zone.

### DateTime format

| Format  | Display in DateTimePicker            |
|---|--------------------------------------|
| Short Date and Time – "d/M/yy h:mm tt"                    | 9/12/2014 2:04 PM                    |
| Medium Date-d MMM yy h:mm tt                              | 12 Sep 14 2:04: PM                   |
| Full Date and short time - dddd, MMMM dd, yyyy HH:mm tt   | Friday, September 12,2014 2:04 PM    |
| Full Date and Long Time - dddd, MMMM dd, yyyy HH:mm:ss tt | Friday, September 12,2014 2:04:00 PM |
| UTC - yyyy-MM-dThh:mm:ssz                                 | 2014-09-12T2:04:00+5                 |

You can also customize the format. Refer the following list to create your custom format for **DateTimePicker**.



- d - Day of the month.
- ddd - Short name of day of the week.
- dddd - Full name of day of the week.
- M – The month, from 1 through 12.
- MMM- Short name of Month.
- MMMM- Long name of the Month.
- yy - Last two digit if year.
- yyyy - Full Year.
- hh – Hour.
- mm – Minutes.
- ss – Seconds.
- tt - The AM/PM designator.

In the following example, set **dateTimeFormat** to **full datetime** format.

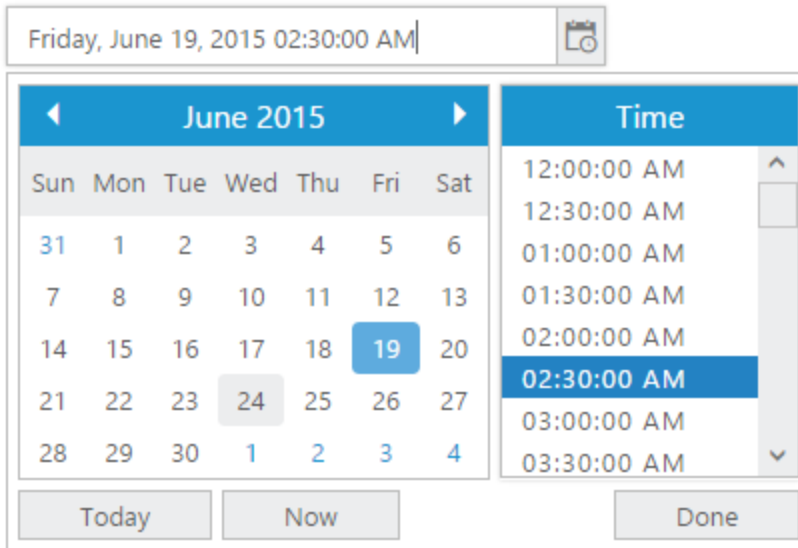
Add the following code in your **HTML** page.

#### HTML

```
<div class="control">
  <input type="text" id="dateTime" />
</div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateTimePickerComponent {
  $(function () {
    // Add the code in your script section to render the DateTimePicker
    var datetimeSample = new ej.DateTimePicker($("#datetimepick"), {
      dateTimeFormat: "dddd, MMMM dd, yyyy hh:mm:ss tt",
      timePopupWidth: "150px",
      timeDisplayFormat: "hh:mm:ss tt",
      width: '300px'
    });
  });
}
```



### Day Header format

You can change the format for the days of the week names using **Day Header format** property. By default in our **DateTimePicker** day of the week format in **showHeaderMin** format. For example, **Sun** for **Sunday**. To know the different types of day format refer the following table.

showHeaderMin

| Header Format types | Description  |
|---------------------|--|
| showHeaderNone      | Removes the day header   |
| showHeaderShort     | Shows the day header format in min like Su, Mo, Tu â€¦               |
| showHeaderMin       | Shows the day header format in short like Sun, Mon, Tue â€¦          |
| showHeaderLong      | Shows the day header format in long like Sunday, Monday, Tuesday â€¦ |

You can also customize the format according to your needs. This is achieved by changing the day names information in the culture script file. This is explained later under the Localization section of this document. In the following sample is displayed, the short name of the day of the week, by setting day header format as **showHeaderShort**.

Add the following code in your **HTML** page.

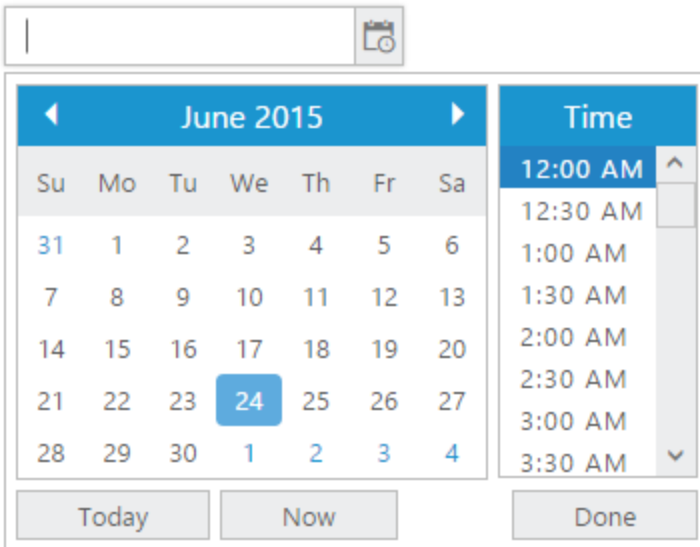
### HTML

```
<div class="control">
  <input type="text" id="dateTime" />
</div>
```

### JAVASCRIPT

```
// Add the code in your script section to render the DateTimePicker with
short header
var datetimeSample = new ej.DateTimePicker($("#datetimepick"), {
```

```
dayHeaderFormat: "showHeaderShort",
width: '200px',
});
```



### Date Range

**Date Range** between two dates is achieved by **minDateTime**, **maxDateTime** property of **DateTimePicker**.

In **DateTimePicker**, a property called **enableStrictMode** is available. When this property is set to false, it does not allow wrong values and corrects the entered value automatically. If it is true, then **DateTimePicker** allows values with error class to indicate the selected date is wrong.

**minDateTime** - Sets the minimum value to the **DateTimePicker**. Based on the **enableStrictMode** value behind the minimum value an error class is added to the wrapper element or it is corrected automatically to the nearest correct date value.

**maxDateTime** - Sets the maximum value to the **DateTimePicker**. Based on the **enableStrictMode** value beyond the maximum value an error class is added to the wrapper element or it is corrected automatically to the nearest correct date value.

Sometimes, you can give restrictions on selecting the date before or after the particular date. Consider you are going to make a project for hotel reservation system. The “**In DateTime**” has to be lesser than the “**Out DateTime**” and vice versa. So you have to set “**In DateTime**” as minimum **DateTime** and “**Out DateTime**” as maximum **DateTime** for selection in **DateTimePicker** control.

The dates before min date and after the max date are considered as invalid dates and it is disabled for selection.

In the following example September 10, 2014 2.00 PM is set as **minDateTime** and September 21, 2014 2.00 PM set as **maxDateTime**.

Add the following code in your **HTML** page.

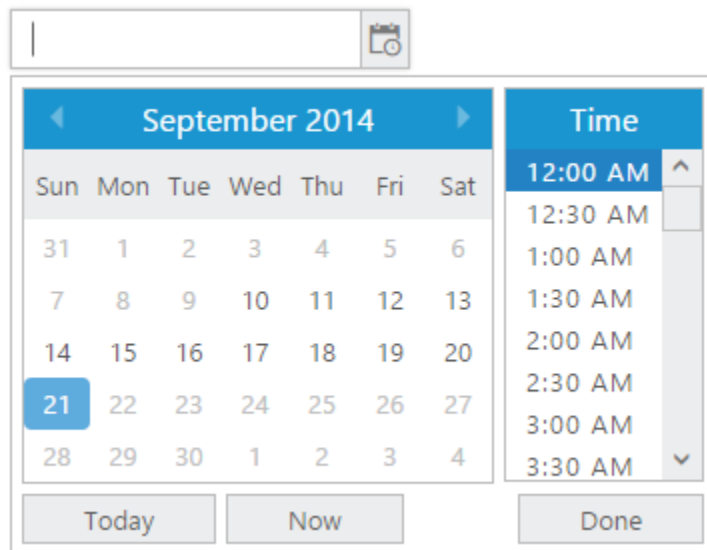
### HTML

```
<div class="control">
```

```
<input type="text" id="dateTime" />
</div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateTimePickerComponent {
$(function () {
// Add the code in your script section to render the DateTimePicker with
fixed minimum and maximum date and time
var datetimeSample = new ej.DateTimePicker($("#datetimepick"), {
width: 200,
minDateTime: "9/10/2014 2:00 PM",
maxDateTime: "9/21/2014 2:00 PM",
});
});
}
```



## Start and Depth level

### Start Level

**DateTimePicker** allows you to change the starting level view of Calendar inside the **DateTimePicker** popup. Consider you are creating a login form for your blog. If the “Birth date” field in the form starts from year, it is easy to select date from year, month and date. By default, the Start Level of **DateTimePicker** is “Month” level view. Refer the following table to know the different types of start level.

### Start Level

| Start Level | Syntax              | Description                   |
|-------------|---------------------|-------------------------------|
| month       | startLevel: “month” | Starts from month level view. |
| year        | startLevel: “year”  | Starts from year level view.  |

|         |                       |                                     |
|---------|-----------------------|-------------------------------------|
| decade  | startLevel: "decade"  | Starts from year decade level view. |
| century | startLevel: "century" | Starts from century level view.     |

In the following example **DateTimePicker** popup start level view is set to "century level" when you drop down the **DateTimePicker** popup.

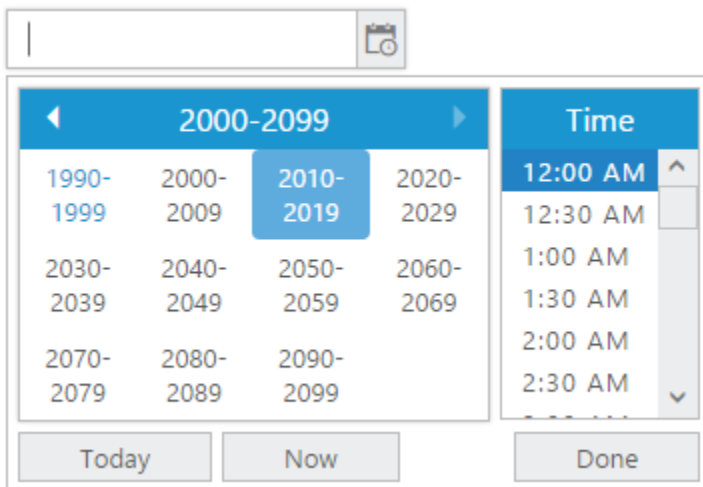
Add the following code in your **HTML** page.

### HTML

```
<div class="control">
<input type="text" id="dateTime" />
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateTimePickerComponent {
$(function () {
// Add the code in your script section with start level as century in
DateTimePicker popup
var datetimeSample = new ej.DateTimePicker($("#datetimepick"), {
startLevel: "century",
width: '200px',
});
});
}
```



### Depth Level

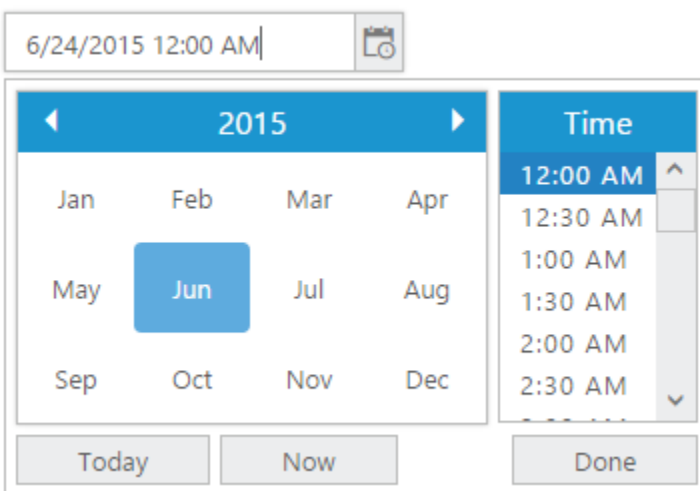
**DateTimePicker** enables you to set the drill down level of **DateTimePicker**. Consider, you are going to take the monthly report of sales in your company. In that case, there is no need for selecting date from month view. You can restrict the drill-down of **DateTimePicker** popup to month view.

In the following example, start level is set from century and drill down is set to year view. So drill-down to month is restricted. So you can avoid selecting the day of the month. By default, the current day of the month is maintained in the day number field. Refer the following code to set drill down value to year view.

The following code snippet is set to depth level in **DateTimePicker**.

### JAVASCRIPT

```
var datetimeSample = new ej.DateTimePicker($("#datetimepick"), {
  startLevel: "century",
  depthLevel: "year",
  width: '200px',
});
```



### Date in other months

**DateTimePicker** calendar can display the dates in other months at the start or end of the current month. To enable or disable the display of other month dates in the current month, you can use the property called **showOtherMonths**. By setting this property value as **"true"** you can display the dates in other months at the start or end of the current month. By default the value of this property is **"true"**.

Consider you are going to calculate the monthly report of your company's employee attendance. To avoid the mistake of selecting other month dates while calculating current month report, you can disable showing other month dates in the current month. You can achieve this requirement by setting **showOtherMonths** value as **"false"**.

Add the following code in your **HTML** page.

### HTML

```
<div class="control">
  <input type="text" id="datetime" />
</div>
```

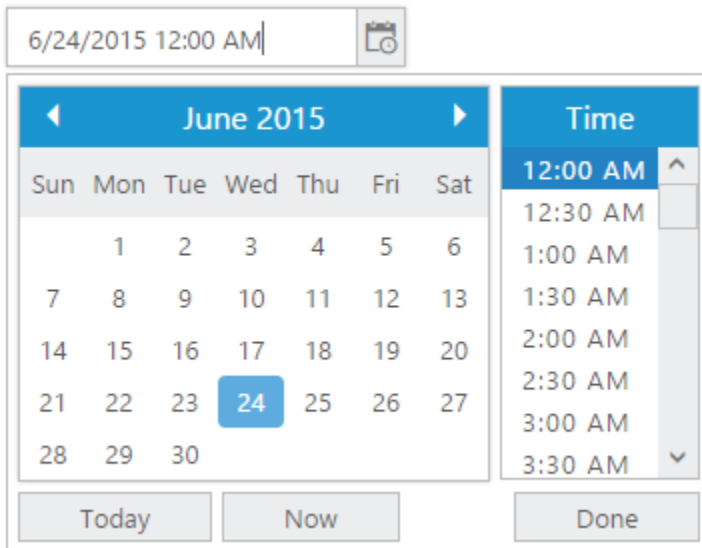
### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```

module DateTimePickerComponent {
$(function () {
// Add the code in your script section to render DateTimePicker without
displaying other month dates in current month
var datetimeSample = new ej.DateTimePicker($("#datetime"), {
showOtherMonths: false,
width: '200px',
});
});
}

```



### Time Interval

You can set time interval between two adjacent time values in the time popup manually using the **interval** property. By default the value of the **interval** property is 30 minutes. Setting this value as 60 is considered as 1 hour. Sometimes you need to update for every hour work log reports. In that case to select time from time popup window with 1 hour interval to update the every 1 hour report you can use **interval** option by setting time interval value as "60 minutes".

Add the following code in your **HTML** page.

#### HTML

```

<div class="control">
<input type="text" id="dateTime" />
</div>

```

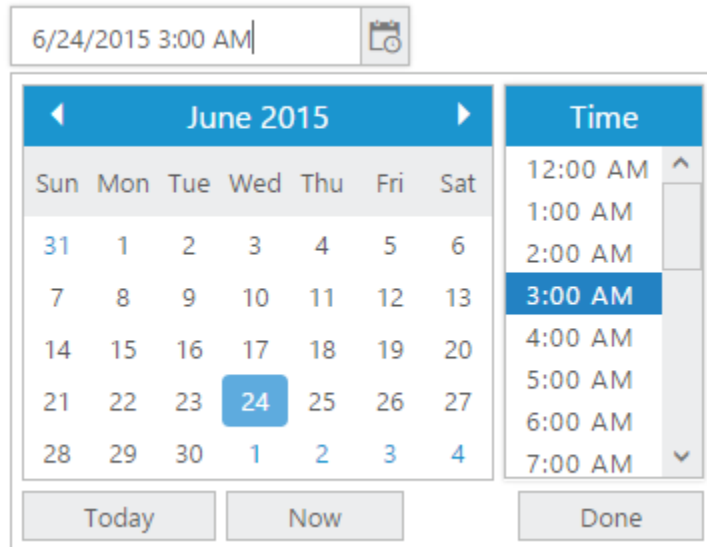
#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateTimePickerComponent {
$(function () {
// Add the code in your script section to render DateTimePicker with 1 hour
interval between two adjacent times in time picker popup
var datetimeSample = new ej.DateTimePicker($("#datetimepick"), {

```

```
interval: 60,
width: '200px',
});
});
}
```



## Globalization

DateTimePicker has been provided with built-in localization support, so that it can adapt based on culture specific locale defined for it.

More than 350 culture specific files are available to localize the datetime. To know more about EJ globalization support, please refer the below link

<https://help.syncfusion.com/js/localization>

**Note:** Seven culture-specific script files are available in the below specified location. For all other culture files, please download from the [GitHub](#) location.

```
(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n
For example, If you have installed the Essential Studio package within C:\Program Files (x86), then
navigate to the below location, C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n
```

To translate our control content from default English to any of the culture, say For example - Spanish language, then you need to refer the ej.culture.es-ES.min.js file in your application,

The **en-US** locale is currently being used as default culture in DateTimePicker. You can set any other culture to DateTimePicker using **Locale** property. Below code example shows Spanish cultured DateTimePicker.

Refer the below Spanish culture file in head section of HTML page after the reference of **ej.web.all.min.js** file.

## JAVASCRIPT



```
<script src="https://cdn.syncfusion.com/js/assets/i18n/ej.culture.es-ES.min.js"></script>
```

If you want to change month names to your culture month just replace month names with your culture month names or your customized format.

The following code example is used to set DateTimePicker in Spanish language.

#### JAVASCRIPT

```
calendars: {
  standard: {
    firstDay: 1,
    days: {
      names: ["domingo", "lunes", "martes", "miércoles", "jueves", "viernes", "sábado"],
      namesAbbr: ["do.", "lu.", "ma.", "mi.", "ju.", "vi.", "sá."],
      namesShort: ["D", "L", "M", "X", "J", "V", "S"]
    },
    months: {
      names: ["enero", "febrero", "marzo", "abril", "mayo", "junio", "julio", "agosto", "septiembre", "octubre", "noviembre", "diciembre", ""],
      namesAbbr: ["ene.", "feb.", "mar.", "abr.", "may.", "jun.", "jul.", "ago.", "sep.", "oct.", "nov.", "dic.", ""]
    },
    AM: null,
    PM: null,
    eras: [{
      "name": "d. C.",
      "start": null,
      "offset": 0
    }],
    patterns: {
      d: "dd/MM/yyyy",
      D: "dddd, d' de 'MMMM' de 'yyyy",
      t: "H:mm",
      T: "H:mm:ss",
      f: "dddd, d' de 'MMMM' de 'yyyy H:mm",
      F: "dddd, d' de 'MMMM' de 'yyyy H:mm:ss",
      M: "d' de 'MMMM",
      Y: "MMMM' de 'yyyy"
    }
  }
}
```

The following code example can be used to get Spanish culture in **DateTimePicker**.

Add the following code in your **HTML** page.

#### HTML

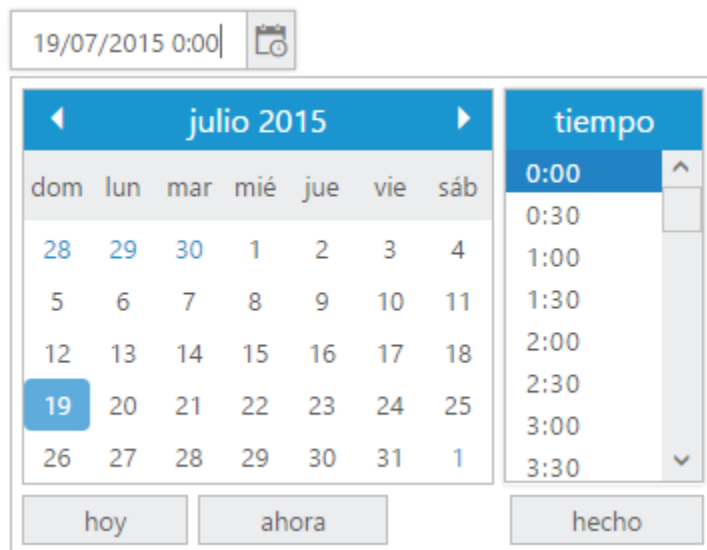
```
<div class="control">
  <input type="text" id="dateTime" />
</div>
```

#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateTimePickerComponent {
$(function () {
// Add the code in your script section to render the DateTimePicker with
Spanish culture
var datetimeSample = new ej.DateTimePicker($("#datetimepick"), {
locale: "es-ES",
buttonText: { today: "hoy", timeNow: "ahora", done: "hecho", timeTitle:
"tiempo" },
});
});
}

```



### Right-to-Left

RTL control supports right-to-left functionality and features for languages that work right-to-left for selecting and editing date and time. You can change your popup and textbox display to read right-to-left. Arabic and Hebrew are written from right to left. The customer who has a right to left writing style can use this feature. You can achieve this in your **DateTimePicker** by using **enableRTL** property. Setting this property to **"true"** allows you to write in the right to left format. Position of the toolbars are also changed to right to left.

Add the following code in your **HTML** page.

#### HTML

```

<div class="control">
<input type="text" id="datetime" />
</div>

```

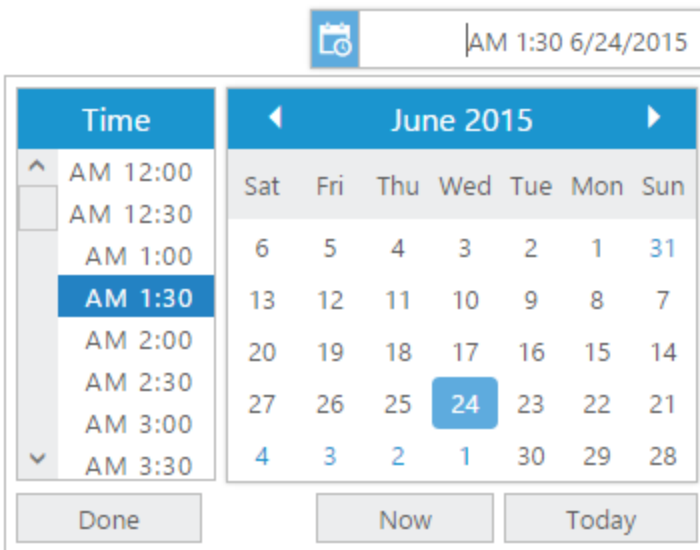
#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateTimePickerComponent {
$(function () {

```

```
// Add the code in your script section to render DateTimePicker with right
to left appearance
var datetimeSample = new ej.DateTimePicker($("#datetime"), {
width: 200,
enableRTL: true
});
});
}
```



## Appearance and Styling

### Theme

**DateTimePicker** control support rich appearance. This control consist of six flat themes and six gradient themes. To use these twelve themes, refer the themes files in HTML file.

You need two style sheets to apply styles to **DateTimePicker** control; one **ej.widgets.core.min.css** and one **ej.theme.min.css**. If you use **ej.widgets.all.min.css** then you don't need to use **ej.widgets.core.min.css** and **ej.theme.min.css** because **ej.widgets.all.min.css** is a combination of these two.

The core style sheet applies styles related to positioning and size, but are not related to the color scheme and always require the control to look correct and function properly. The theme styles sheet applies theme-specific styles like colors and backgrounds.

The following list is of the twelve themes supported by **DateTimePicker**:

- default-theme
- flat-azure-dark
- flat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark

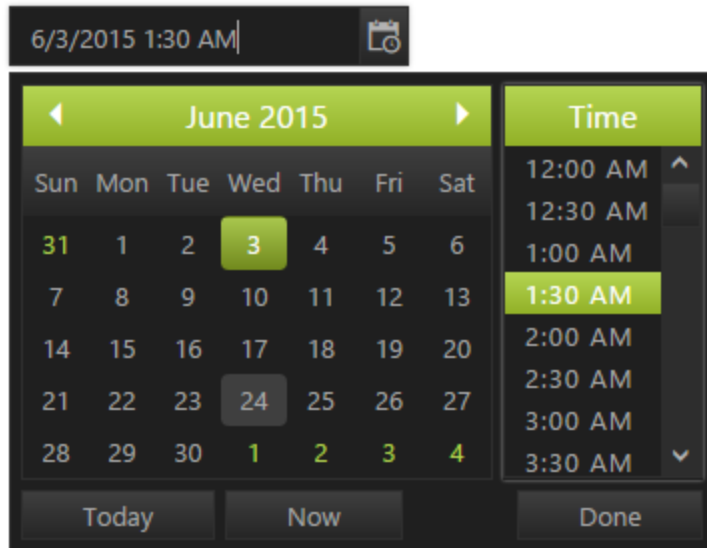
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

Add the following code in your **HTML** page.

### HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/gradient-lime-dark/ej.web.all.min.css" rel="stylesheet" />
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"> </script>
</head>
<body>
<div class="control">
<input type="text" id="dateTime" />
</div>
<script>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DateTimePickerComponent {
$(function () {
// declaration
var datetimeSample = new ej.DateTimePicker($("#datetimepick"), {
width: '200px'
});
});
}
</script>
</body>
</html>
```

**Note:** jQuery.easing external dependency has been removed from version 14.3.0.49 onwards. Kindly include this jQuery.easing dependency for versions lesser than 14.3.0.49 in order to support animation effects.



### CSS Class

**DateTimePicker** control also allows you to customize its appearance using user-defined CSS and custom skin options such as colors and backgrounds. To apply custom themes you have a property called **cssClass**. **cssClass** property sets the root class for **DateTimePicker** theme.

Using this **cssClass** you can override the existing styles under the theme style sheet. The theme style sheet applies theme-specific styles like colors and backgrounds. In the following example, the value of **cssClass** property is set as “**Purple-dark**”. **Purple-dark** is added as root class to **DateTimePicker** control at the runtime. From this root class you can customize the **DateTimePicker** control theme.

Add the following code in your **HTML** page to render the DateTimePicker.

### HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/flat-azure-dark/ej.web.all.min.css" rel="stylesheet" />
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-1.10.2.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/ej.web.all.min.js"> </script>
</head>
<body>
<div class="control">
<input type="text" id="dateTime" />
</div>
<script>
$(function () {
// Add the code in your script section to render the DateTimePicker with
cssClass property
var datetimeSample = new ej.DateTimePicker($("#datetimepick"), {
width: 200,
cssClass: "Purple-dark"
});
});
```

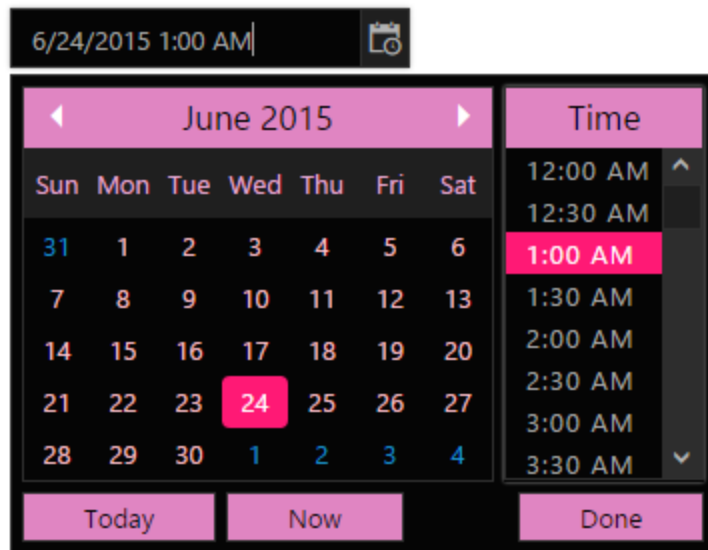
```
</script>
</body>
</html>
```

In the following style sheet the exiting theme style sheet file has been over-ridden using root class **"Purple-dark"**.

Add the following code in your style section.

### CSS

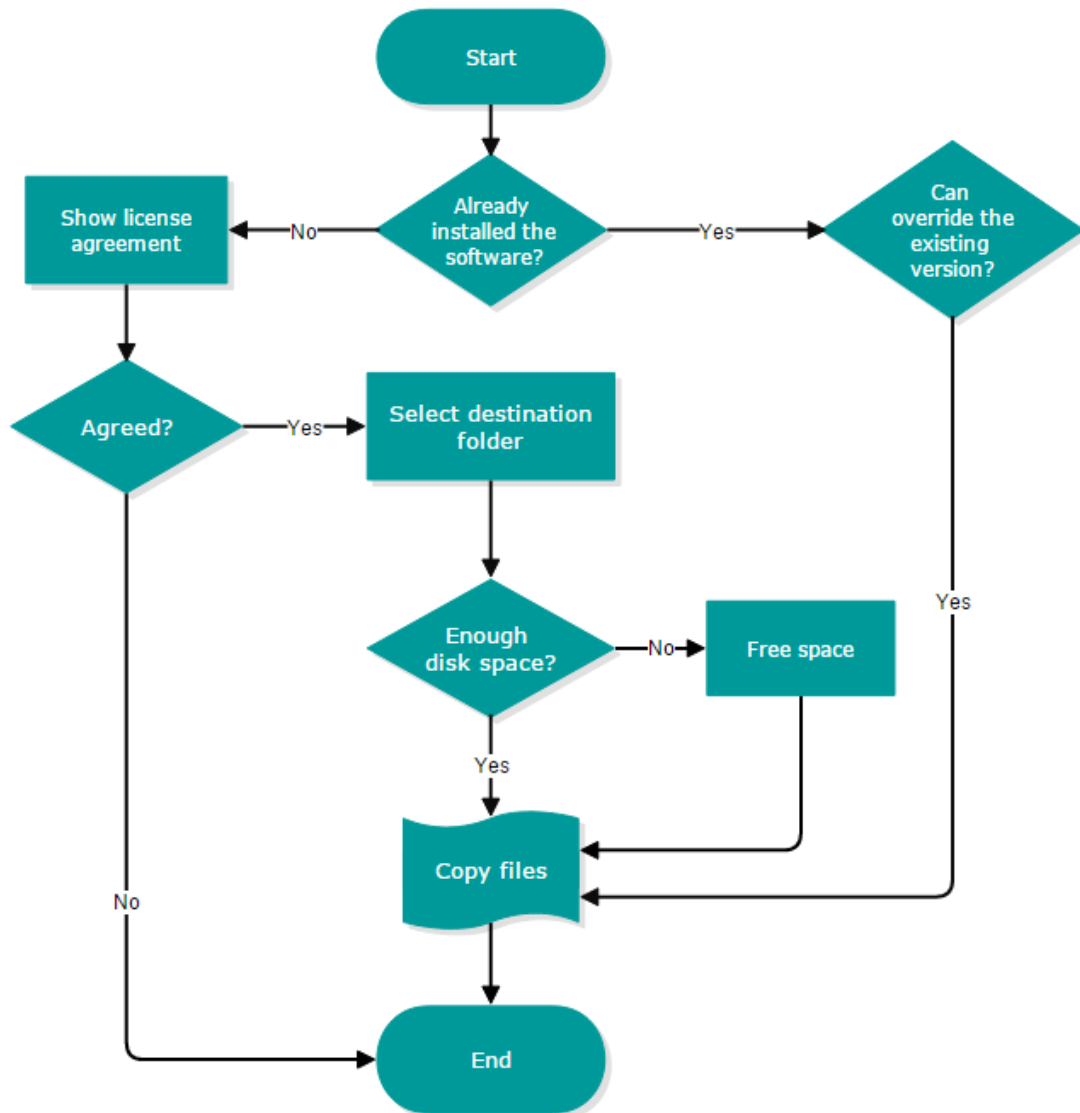
```
<style>
.Purple-dark .e-week-header {
color: #EBADD6;
}
.Purple-dark .e-text {
color: black;
}
.Purple-dark .e-state-default {
color: pink;
}
.Purple-dark .e-active {
background-color: #FF1975;
}
.Purple-dark .e-state-default:hover {
color: #EBADD6;
}
.Purple-dark .e-dt-button {
color: black;
background-color: #E085C2;
}
.Purple-dark .e-header {
background-color: #E085C2;
color: black;
}
.Purple-dark .e-timecontainer .e-header {
background-color: #E085C2;
color: black;
}
.Purple-dark .e-datepicker table td:hover,
.Purple-dark .e-datepicker td.e-state-hover,
.Purple-dark .e-datepicker .current-month.e-state-default.e-special-
day:hover {
background-color: #FF1975;
}
.Purple-dark .e-timewidget .e-select:hover,
.Purple-dark .e-time-popup.e-popup .e-hover {
background: #FF1975;
color: white;
}
.Purple-dark .e-datetime-wrap:hover {
background: #FF1975;
}
</style>
```



## Diagram

### Overview

**Essential Diagram TypeScript** creates rich Visio-like applications. Its Framework comprises of many Elements that helps you to create an application easily. The rich feature set of the Diagram control includes Snapping, Guidelines, Gridlines, Serialization and Zooming.



The list of rich features of Diagram control in React JS is as follows.

- **Node, Connector, Group, Port, Label:** Element used to compose diagram.
- **Symbol Palette:** It holds a list of symbols that is dropped over diagram.
- **Clipboard Commands:** Performs Cut, Copy and Paste operations.
- **Undo/Redo:** Performs correction in recent change.
- **Serialization:** Save current state of Diagram, and load them back when needed.
- **Snapping:** Snap the diagram elements towards the nearest elements.
- **Gridlines:** Visual horizontal/vertical lines that helps to align elements on diagram.
- **Interaction:** Zoom, pan, multiple selections, keyboard shortcuts, snapping.
- **Layout:** Arranges nodes in a tree-like structure based on relationship between the Nodes.

### Getting started

This section explains briefly about how to create a **Diagram** control in your application with **TypeScript**.



### Adding Script Reference

Create an **HTML** page and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/14.3.0.49/js/web/bootstrap-
theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
<script
src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.validate.
min.js"></script>
<script src="http://js.syncfusion.com/demos/web/scripts/xljsondata.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/14.3.0.49/js/web/ej.web.all.min.js"
type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

In the above code, `ej.web.all.min.js` script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

### Initialize Diagram

Add a `div` container to render the Diagram.

#### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="diagram"></div>
</body>
</html>
```

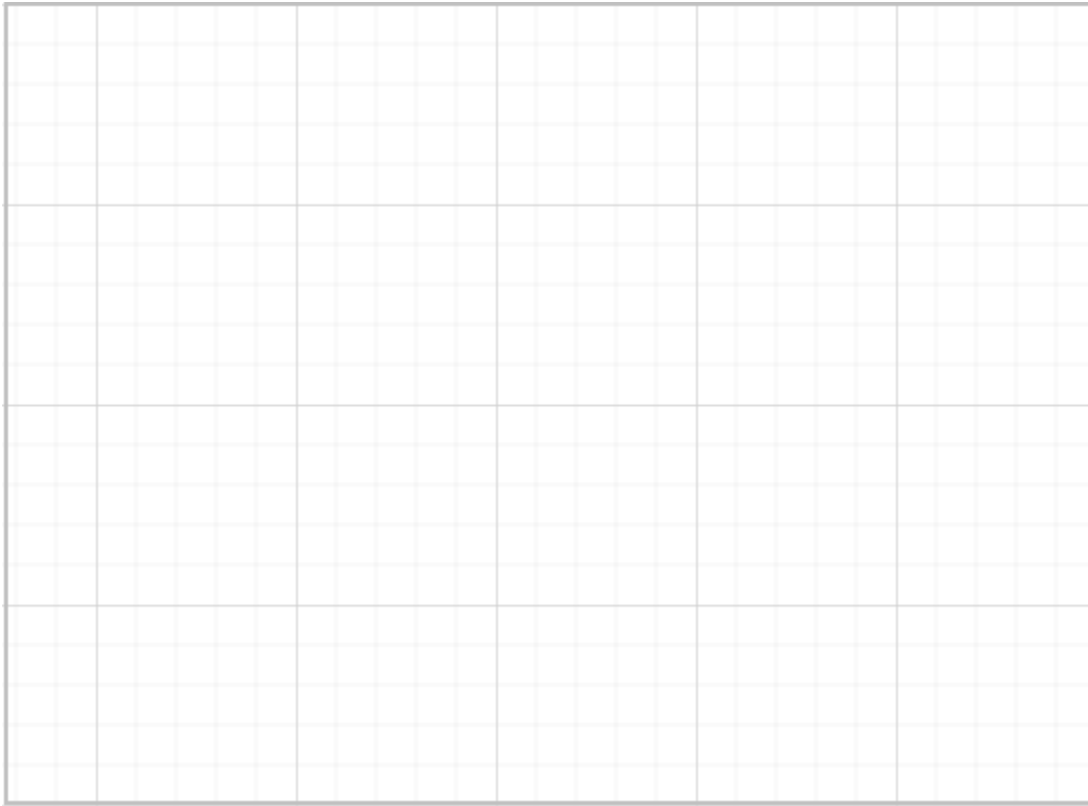
Initialize the Diagram in ts file by using the `ej.diagram` method. The Diagram is rendered based on default `width` and `height`.

#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
$(function () {
var diagram = new ej.datavisualization.Diagram($("#diagram"), {
width: "1000px",
height: "600px"
```

```
});  
});
```

This creates an empty diagram as shown in image



Populate Diagram with nodes and connectors

- This section explains how to populate JSON data to the Diagram.
- The Diagram is rendered based on default **width** and **height**. You can also customize the Diagram dimension by setting the **width** and **height** attribute in **scrollSettings**.

## HTML

```
<!DOCTYPE html>  
<html>  
<body>  
<script type="text/javascript" src="diagram/diagram.js">  
</script>  
<div id="diagram"></div>  
</body>  
</html>
```

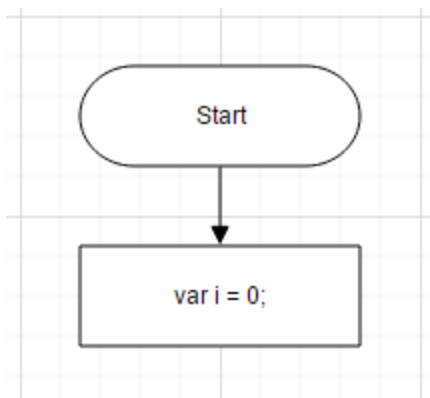
## JAVASCRIPT

```
$(function () {  
  var diagram = new ej.datavisualization.Diagram($("#diagram"), {  
    width: "1000px",
```

```

height: "600px",
nodes: [
createNode({ name: "Start", offsetX: 300, offsetY: 50, width: 140, height:
50, labels: [{ text: "Start" }], type: "flow", shape: "terminator"}),
createNode({ name: "Init", offsetX: 300, offsetY: 140, width: 140, height:
50, labels: [{ text: "var i = 0;" }], type: "flow", shape: "process" }),
],
connectors: [
createConnector({ name: "connector1", sourceNode: "Start", targetNode:
"Init" }),
]
});
});
function createNode(option: ej.datavisualization.Diagram.Node) {
return option;
}
function createConnector(option: ej.datavisualization.Diagram.Connector) {
return option;
}
function createLabel(options : any) {
return options;
}

```



## Dialog

### Overview

The Typescript Dialog control helps to display a pop-up window within a web page. The Dialog enables a message to be displayed, such as supplementary content like images and text, and an interactive content like forms.

### Key Features

**Model dialog support:** Displays the content in a modal dialog, disabling interaction with other items on the page. **AJAX Load:** Load AJAX content in dialog content panel. **Drag support:** Drag the Dialog within the page. **Customized dialog position:** By default, the dialog is shown in the center of the container. If given a position, the dialog is displayed at given position. \* **Keyboard navigation:** Users can interact with the dialog by using the keyboard.

### Getting Started

This section helps you to understand the getting started of the Dialog control with the step-by-step instructions.

## Create a Dialog

The following steps guide you to add a Dialog control.

1) Refer the common Typescript [Getting Started Documentation](#) to create an application and add necessary scripts and styles for rendering the Dialog control. 2) Create an HTML page and add the scripts and css references in the order mentioned in the following code example.

### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Typescript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.web
.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add Dialog code here-->
</body>
</html>
```

3) Add the following code with in the tag to render the Dialog control.

### HTML

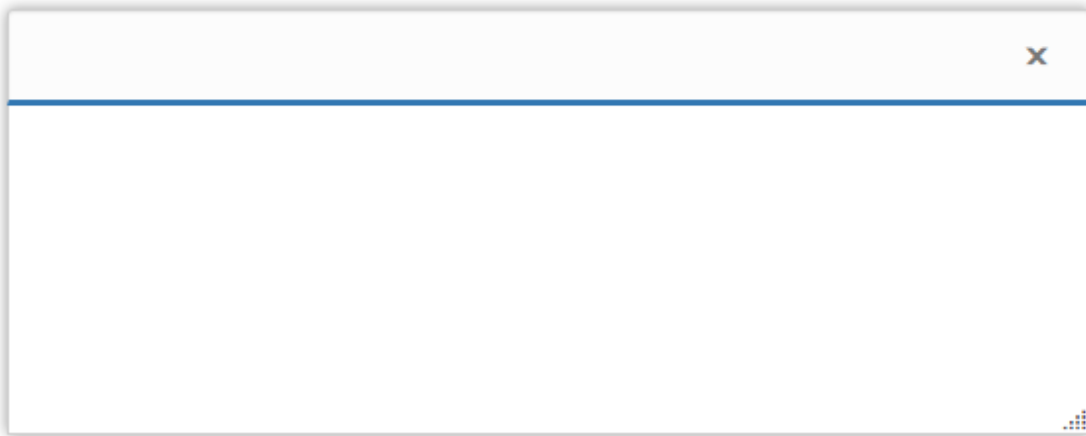
```
<div id="basicDialog">
</div>
```

Initialize the Dialog control in ts file by using the ej.Dialog method.

### TS

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module DialogComponent {
$(function () {
var dialogInstance = new ej.Dialog($("#basicDialog"), {
});
});
}
```

To get the following output from the above-mentioned code.



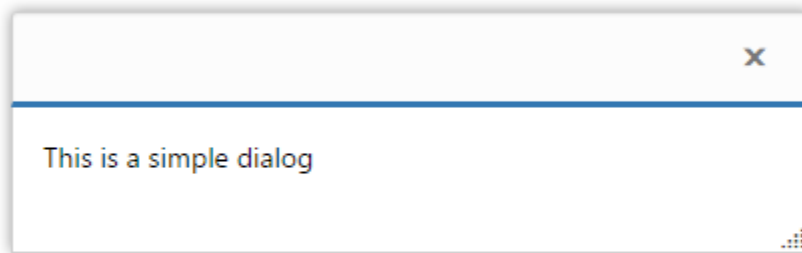
### Add dialog content

Add the below code to render dialog control with content.

#### HTML

```
<div id="basicDialog">  
<p>This is a simple dialog</p>  
</div>
```

To get the following output from the above-mentioned code



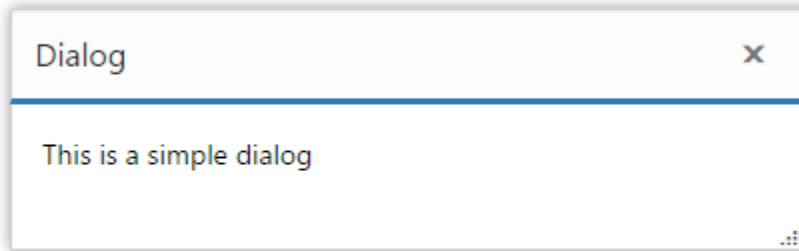
### Set the title

To set the dialog control title as using the below mentioned code.

#### HTML

```
<div id="basicDialog" title="Dialog">  
<p>This is a simple dialog</p>  
</div>
```

To get the following output from the above-mentioned code.



### Open Dialog dynamically

In most cases, the Dialog controls are needed only in dynamic actions like showing some messages on clicking a button, to provide alert, etc. So, the Dialog control provides “open” and “close” methods to open/close the dialogs dynamically.

The Dialog control can be hidden on initialize using showOnInit property which should be set to false.

The dialog will be opened on clicking the Button control. Refer to the below mentioned code.

### HTML

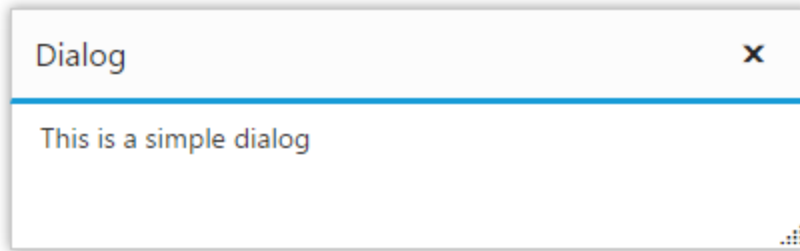
```
<input class="e-btn" id="btnOpen" value="Click to open dialog" />
<div class="control">
  <div id="basicDialog" title="Dialog">
    <p>This is a simple dialog</p>
  </div>
</div>
</div>
```

### JS

```
module DialogComponent {
  $(function () {
    var dialogInstance = new ej.Dialog($("#basicDialog"), {
      target:".control",
      showOnInit: false,
      close:()=>{
        $("#btnOpen").show();
      }
    });
    var btnInstance = new ej.Button($("#btnOpen"), {
      size: "medium",
      click: ()=>{
        $("#btnOpen").hide();
        $("#basicDialog").ejDialog("open");
      },
      type: "button",
    });
  });
}
```

To get the following output from the above-mentioned code.

Click to open Dialog



**Note:** You can find the Dialog control properties from the [API reference](#).

### Load content

By default, the content inside the Dialog element is considered as the content for the Dialog widget.

Also, we can render the Dialog widget content through the following ways.

1. request through AJAX
2. request iframe content
3. request image content

This settings can be specified through `contentType` property.

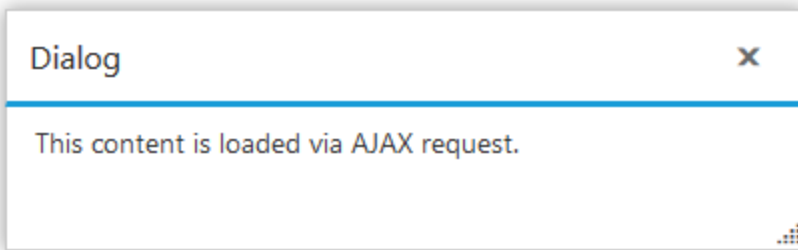
### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module DialogComponent {
  $(function () {
    var dialogInstance = new ej.Dialog($("#basicDialog"), {
      title: "Dialog",
      //create a HTML file (dialogcontent.html) which contains the content for the
      //dialog
      contentUrl: "dialogcontent.html",
      contentType: "ajax"
    });
  });
}
```

Here below the content of that HTML file.

### HTML

```
<div id="content">
  This content is loaded via AJAX request.
</div>
```



We can handle the AJAX request's success and failures through the events "ajaxSuccess" and "ajaxError" events respectively. See also ajaxSuccess and ajaxError

The previous example is modified as below to handle the success and failure events.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DialogComponent {
$(function () {
var dialogInstance = new ej.Dialog($("#dialog"), {
title: "Dialog",
//create a HTML file (dialogcontent.html) which contains the content for the
dialog
contentUrl: "dialogcontent.html",
contentType: "ajax",
ajaxSuccess: "onSuccess",
ajaxError:"onError"
});
});
}
function onSuccess(args) {
//handle success event
}
function onError(args) {
//handle success event
}
```

**Note:** The same way we can render the iframe and image content for the Dialog widget by specifying the `contentType` as "iframe" and "image" respectively and also by specifying the proper location in the `contentUrl` property.

### Action Buttons

The Dialog widget provides the following action buttons.

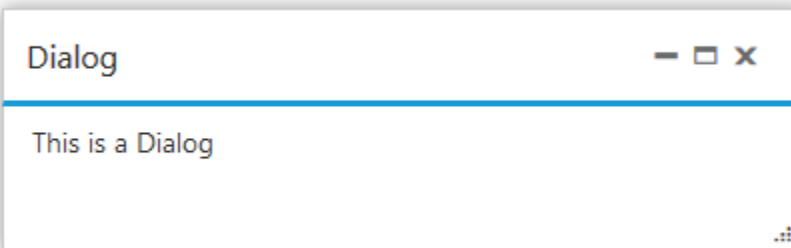
1. Close
2. Maximize
3. Minimize
4. Pin/Unpin
5. Collapse/Expand



You can display only the necessary buttons in the Dialog widget by configuring the `actionButtons` property.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DialogComponent {
  $(function () {
    var dialogInstance = new ej.Dialog($("#dialog"), {
      showOnInit: false,
      actionButtons: ["close", "maximize", "minimize"]
    });
  });
}
```



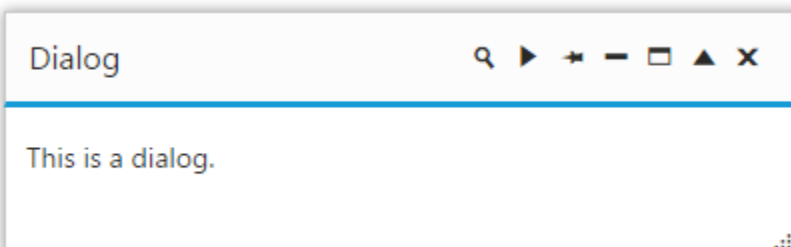
#### Customizing Action Buttons

We can customize the action buttons in dialog widget.

You can add new action button in the dialog widget by configuring the `actionButtonClick` event.

#### JAVASCRIPT

```
//create dialog widget
var dialogInstance = new ej.Dialog($("#dialog"), {
  showOnInit: false,
  actionButtons: ["close", "collapsible", "maximize", "minimize", "pin",
    "mediaplay", "search"],
  actionButtonClick: "playMedia"
});
function playMedia(args)
{
  console.log(args.buttonID);
}
```



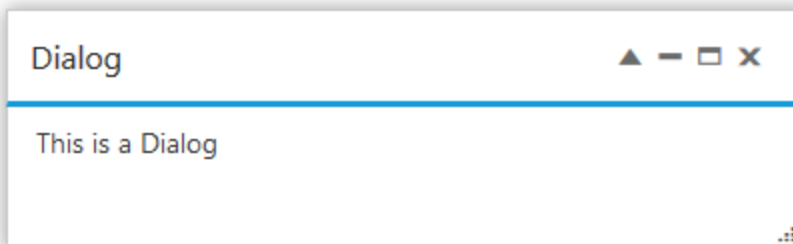
## Animation

The Dialog widget can be animated while opening and closing the dialog.

We can specify the effect and the duration for the animation. There are two types of effects. They are slide and fade. We can configure these settings separately for open and close dialog actions.

### JAVASCRIPT

```
//create dialog widget
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DialogComponent {
$(function () {
var dialogInstance = new ej.Dialog($("#dialog"), {
showOnInit: false,
actionButtons: ["close", "maximize", "minimize", "collapsible"],
enableAnimation: true,
animation: {
//animation settings while opening the dialog
show: {
effect: "slide",
duration: 500
},
//animation settings while closing the dialog
hide: {
effect: "fade",
duration: 500
}
}
});
});
}
```



## Keyboard interaction

You can use Keyboard shortcut keys as an alternative for mouse actions to interact with the Dialog widget. The keyboard interaction will be enabled by default in the Dialog widget. It can be disabled by setting `allowKeyboardNavigation` to false. Please refer the below table for details about short cut keys and its corresponding usage.

| Shortcut Key | Usage                                   |
|--------------|---|
| Up           | Moves the dialog at upward direction.   |
| Down         | Moves the dialog at downward direction. |

|            |   |
|------------|---|
| Left       | Moves the dialog at left direction.   |
| Right      | Moves the dialog at right direction.  |
| Ctrl Up    | Reduces the dialog height.  |
| Ctrl Down  | Increases the dialog height.  |
| Ctrl Left  | Reduces the dialog width.   |
| Ctrl Right | Increases the dialog width.   |
| Esc        | Closes the dialog<br>N> It will work only if <code>closeOnEscape</code> is set to true. |

## How To?

### Create Multiple Dialogs

Essential JS library supports multiple Dialog widgets in the same web page with different contents and different functionalities.

Initialize the Dialog widgets by adding the script section as below.

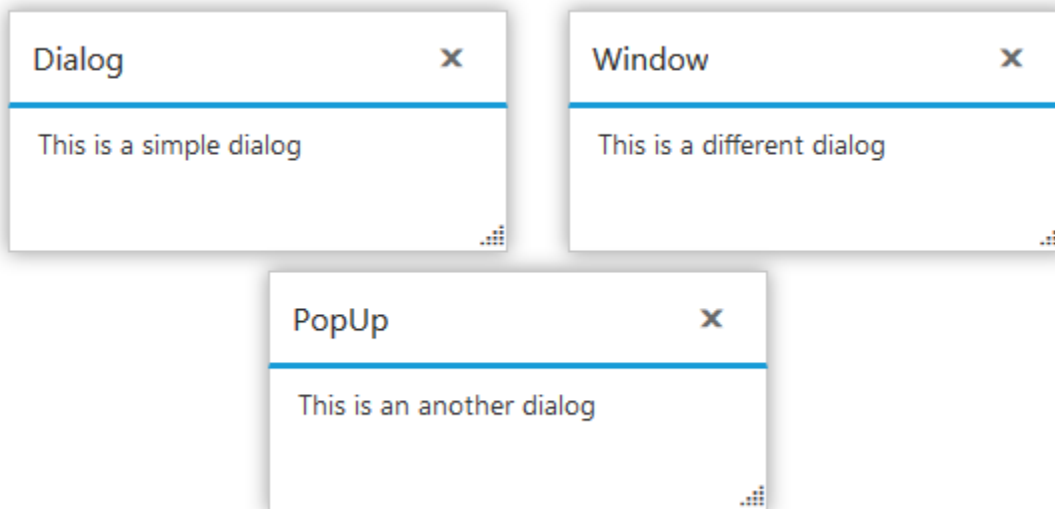
### HTML

```
<!--dialog 1-->
<div id="dialog1" title="Dialog">
<p>
This is a simple dialog
</p>
</div>
<!--dialog 2-->
<div id="dialog2" title="Window">
<p>
This is a different dialog
</p>
</div>
<!--dialog 3-->
<div id="dialog3" title="PopUp">
<p>
This is an another dialog
</p>
</div>
<script>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DialogComponent {
$(function () {
var dialogInstance = new ej.Dialog($("#dialog1"), {
width: 250,
position: {
X: 20,
Y: 20
}
});
var dialogInstance = new ej.Dialog($("#dialog2"), {
```

```
width: 250,
position: {
X: 300,
Y: 20
}
});
var dialogInstance = new ej.Dialog($("#dialog3"), {
width: 250,
position: {
X: 150,
Y: 150
}
});
});
}
</script>
```

**Note:** If the position of the dialog is not set as above, all the three dialogs will be overlapped with each other.

#### [position](#)



#### Create Nested Dialog

A Dialog widget can be nested within another Dialog widget.

Create a div element to render the child Dialog widget and use it as a content of parent Dialog widget.

#### HTML

```
<!--button to open the dialog-->
<button id="button1">Open Dialog</button>
<div id="dialog">
<!-- button to open the nested dialog -->
<button id="button2" class="ejButton">Open Nested Dialog</button>
<!--nested dialog-->
<div id="nesteddialog">
This is a nested Dialog
</div>
```

```
</div>
```

Initialize both the Dialog widgets by adding the script section as below.

### JAVASCRIPT

```
$(function () {
  var buttonInstance = new ej.Button($("#button1"), {
    click: "openDialog"
  });
  var buttonInstance = new ej.Button($("#button2"), {
    click: "openNestedDialog"
  });
  var dialogInstance = new ej.Dialog($("#dialog"), {
    title: "Dialog",
    showOnInit: false,
    width: 500, height: 400,
    actionButtons: ["close", "collapsible", "maximize", "minimize", "pin"]
  });
  var dialogInstance = new ej.Dialog($("#nesteddialog"), {
    title: "Nested Dialog",
    showOnInit: false,
    width: 300, height: 200
  });
});
function openDialog(args) {
  $("#dialog").ejDialog("open");
}
function openNestedDialog(args) {
  $("#nesteddialog").ejDialog("open");
}
```

Create Confirmation Dialog with Footer option.

Essential JS library supports Alert Dialog widgets.

Using `showFooter` property to render Alert Dialog with Footers in Dialog widget.

Initialize the Dialog widget using the below code.

### HTML

```
<div class="content-container-fluid">
  <div class="row">
    <div class="cols-sample-area">
      <input class="e-btn" id="btnOpen" value="Click to open dialog" />
    <div class="control">
      <div id="basicDialog" title="Confirmation Dialog">
        Do you really leave the session?
      </div>
    </div>
  </div>
</div>
```

Initialize Footer in Dialog widgets by adding the script section in JsRender as below.

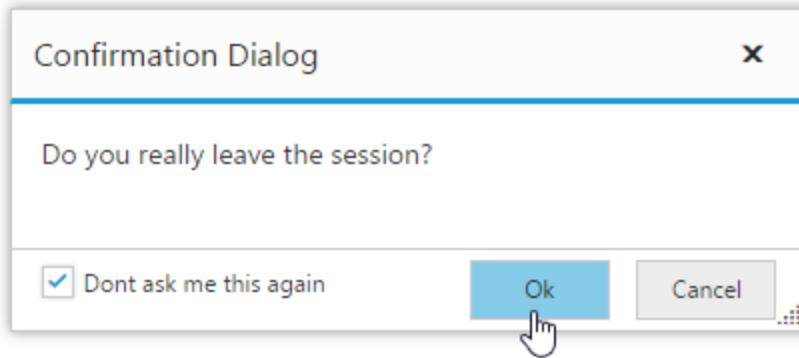
**JAVASCRIPT**

```
<script id="sample" type="text/x-jsrender">
<div class="footerspan" style="float:right">
<button id='btn1'>Ok</button>
<button id='btn2'>Cancel</button>
</div>
<div class="condition" style="float:left; margin-left:15px">
<input type="check" id="check1"></div>
</script>
```

Add the below script to render the Dialog widget.

**JAVASCRIPT**

```
$(window).load(function () {
// declaration
var dialogInstance = new ej.Dialog($("#basicDialog"), {
close: "onDialogClose",
target: ".control",
showFooter: true,
footerTemplateId: "sample"
});
var checkboxInstance = new ej.CheckBox($("#check1"), {
text:"Don't ask me this again"
})
var buttonInstance = new ej.CheckBox($("#btnOpen"), {
size: "medium",
click: "onOpen",
type: "button",
height: 30,
width: 150
});
var buttonInstance1 = new ej.CheckBox($("#btn1"), {
size:"mini",
height: 30,
width: 7
});
var buttonInstance2 = new ej.CheckBox($("#btn2"), {
size:"mini",
height: 30,
width: 70
});
function onDialogClose(args) {
$("#btnOpen").show();
}
function onOpen() {
$("#btnOpen").hide();
$("#basicDialog").ejDialog("open");
}
</script>
```



## DigitalGauge

### Overview

Digital Gauge for Essential Studio encompasses the visualization of the segmented display of both the 7-segment numeric values and the 14-segment alphanumeric characters. It can display the following five kinds of character types:

- Seven Segment
- Fourteen Segment
- Sixteen Segment
- EightCrossEightDotMatrix
- EightCrossEightSquareMatrix

There are several other properties available in Digital Gauge which enable you to perform various customizations such as changing the space between the characters, segment width, segment length, spacing between the segments, and other customizations.

### Key Features

- **Character Types** – Provide support for five different character types.
- **Background Image** - Provide support to add images as the background for the Digital Gauge.
- **Custom Font** - Provide support to enable the custom font for text within the Digital Gauge.

### Getting Started

- This section encompasses the details on how to configure DigitalGauge. Here you will learn how to provide data for a DigitalGauge and display the data in the required way.
- In addition, you will learn how to customize the default DigitalGauge appearance according to your requirements. As a result, you will get a DigitalGauge that shows it as Digital thermometer.
- You can use this DigitalGauge in advertisements, decorative purposes, displaying share details in share market, game score boards, token systems, etc.



### Digital Thermometer

#### Create a Digital Gauge

You can easily create the Digital Gauge widget by using the following steps.

1.First create an TypeScript Project and add the following references in the app.ts

For common getting started of TypeScript , you can refer [here](#).

The default type definition file **ej.web.all.d.ts** needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the intelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the **jquery.d.ts** file in your TypeScript application, which can be downloaded from [here](#).

2.Add the below script reference in the HTML page

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

In the above code, **ej.web.all.min.js** script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

3.Create a

tag.

#### HTML



```
<html> <body> <div id="DigitalGauge"></div> </body> </html>
```

4. Initialize the DigitalGauge in ts file by using the `ej.DigitalGauge` method.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DigitalGaugeComponent {
  $(function () {
    var digitalGaugeSample = new
    ej.datavisualization.DigitalGauge($("#DigitalGauge"));
  });
}
```

Run the above code example and you will get a default Digital Gauge as follows.

text

#### Digital Gauge

##### Set Height and Width values

Basic attributes of each canvas elements are height and width. You can set the height and width of the gauge.

#### JAVASCRIPT

```
$(function () {
  var digitalGaugeSample = new
  ej.datavisualization.DigitalGauge($("#DigitalGauge"), {
    height: 145,
    width: 260,
  });
});
```

Run the above code example and you will see a default gauge with the specified height and width values.

text

#### Digital Gauge with Height and Width

##### Set Items Property

Items have different properties to customize the Digital Gauge.

##### Add Segment and Character Properties

- In the Welcome Board, the text color must be attentive in nature. You can give some segment properties such as segment spacing, segment width, segment color, segment length and segment opacity.

- Character type is to define the Digital representation of the character. The five types of character representation available are,
  1. EightCrossEightDotMatrix
  2. SevenSegment
  3. FourteenSegment
  4. SixteenSegment
  5. EightCrossEightSquareMatrix.

## JAVASCRIPT

```
$(function () {  
  var digitalGaugeSample = new  
  ej.datavisualization.DigitalGauge($("#DigitalGauge"), {  
    height: 145,  
    width: 260,  
    items: [{  
      segmentSettings: { width: 2, length: 20 },  
      characterSettings: { type: "SevenSegment", spacing: 12, },  
      value: "102",  
    }]  
  });  
});
```

Run the above code example and you will see the following output.

A digital gauge displaying the number 102 in a seven-segment font.

## Digital Gauge Segment Properties

### Add Background Image

- Add a  
  
element to set the background for the Digital Gauge.
- Add a style tag in the View page to add the background image for the Digital Gauge.
- Add the required properties to show the background image such as position, margin, display, etc.,

## HTML

```
<div id="frameDiv">  
<div id="DigitalGauge" style="width:100%"></div>  
</div>  
<style>  
#frameDiv {  
  align : center;  
  position : relative;  
  margin : 0px auto;  
  display :table;  
  background-image :url("script/frame.png");
```

```
background-repeat :no-repeat;
}
</style>
```

Run the above code example and you will see the following output.



Digital Gauge Background Image

#### [Add Location](#)

The Location property is used to position the digital letters inside the canvas element.

#### JAVASCRIPT

```
$(function () {
  var digitalGaugeSample = new
  ej.datavisualization.DigitalGauge($("#DigitalGauge"), {
    height: 145,
    width: 260,
    items: [{
      //For Displaying Fahrenheit value
      segmentSettings: { width: 2, length: 20 },
      characterSettings: { type: "SevenSegment", spacing: 12, },
      value: "102", position: { x: 15, y: 40 }
    }]
  });
});
```

Run the above code example and you will see the following output.



Digital Gauge with Segment Location

#### [Add Items Collection](#)

You can further add the Items Collection to display the temperature value like Digital Thermometer.

#### JAVASCRIPT

```
$(function () {
  var digitalGaugeSample = new
  ej.datavisualization.DigitalGauge($("#DigitalGauge"), {
    height: 145, width: 260,
    items: [{
      //For Displaying Fahrenheit value
      segmentSettings: { width: 2, length: 20, spacing: 0 },
```

```

characterSettings: { type: "SevenSegment", spacing: 12, },
value: "102",
position: { x: 15, y: 40 }
},
{
  //For displaying degree symbol
  segmentSettings: { width: 2, length: 5, spacing: 0 },
  characterSettings: { type: "SevenSegment", spacing: 5, },
  value: "°",
  position: { x: 70, y: 28 }
},
{
  //For displaying Fahrenheit symbol
  segmentSettings: { width: 2, length: 20, spacing: 0 },
  characterSettings: { type: "SevenSegment", spacing: 12, },
  value: "F",
  position: { x: 170, y: 40 }
},
{
  //For displaying Celcius value
  segmentSettings: { width: 1, length: 9, spacing: 0, color: "#F5b43f" },
  characterSettings: { type: "SevenSegment", spacing: 12, },
  value: "38",
  position: { x: 70, y: 90 },
},
{
  //For displaying degree symbol
  segmentSettings: { width: 1, length: 3, spacing: 0, color: "#F5b43f" },
  characterSettings: { type: "SevenSegment", spacing: 12, },
  value: "°",
  position: { x: 90, y: 80 }
},
{
  //For displaying celcius symbol
  segmentSettings: { width: 1, length: 9, spacing: 0, color: "#F5b43f" },
  characterSettings: { type: "SevenSegment", spacing: 12, },
  value: "C",
  position: { x: 120, y: 90 }
}]
});
});

```

Run the above code example and you will see the following output.



Digital Gauge with Item Collection

## Basic Settings

### Height and Width Customization

The basic customization for any control is to set the dimension. Here dimension refers to two major attributes such as **height** and **width**. The **height** and **width** assigned in the control will render the canvas element in the given size. The code example to set **height** and **width** is as follow.

#### HTML

```
<div id="DigitalGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module DigitalGaugeComponent {
  $(function () {
    // For Digital Gauge rendering
    var digitalGaugeSample = new
    ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
      // For setting height of the canvas element.
      height: 200,
      // For setting width of the canvas element.
      width: 500,
      // For setting text
      value: "Syncfusion"
    })
  });
}
```

Execute the above code examples to render the **DigitalGauge** as follows.



### Responsive Layout

- For any display devices, the control will be rendered based on the space available in that device. For this purpose, **resizing** property is given to the **Digital Gauge** control. The **Digital Gauge** renders with a given value.
- When the browser resize the canvas element checks the dimension with its parent element. If there are any changes in parent dimension, **Gauge** control will changes the dimension based on its parent element change. This feature is enabled by using the property **isResponsive**

#### HTML

```
<div id="DigitalGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {
  // For Digital Gauge rendering
```

```
var digitalGaugeSample = new
ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
  // For setting width of the canvas element.
  width: 800,
  // For enabling resize.
  isResponsive: true,
})
});
```

Execute the above code examples to render the **DigitalGauge** as follows.



### Themes

Themes give the good appearance to the control. There are two types of **Themes** available for **DigitalGauge** as follows

- flat light
- flat dark

### HTML

```
<div id="DigitalGauge1"></div>
```

### JAVASCRIPT

```
$(function () {
  // For Digital Gauge rendering
  var digitalGaugeSample = new
  ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
    // For setting width of the canvas element.
    width: 800,
    // For enabling resize.
    isResponsive: true,
    // For setting theme for digital gauge.
    themes: "flatdark",
    // For setting text
    value: "LOS ANGELS 40 KM"
  })
});
```

Execute the above code examples to render the **DigitalGauge** as follows.



### Frames

#### Inner and Outer Width Customization

Frames are space that enclose the **Digital Gauge**. The inner width of the **Frame** is the distance between the canvas element and the frame. The outer width is the distance from the frame. The code example to set frame's **innerWidth** and **outerWidth** is as follow.

## HTML

```
<div id="DigitalGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module DigitalGaugeComponent {
  $(function () {
    // For Digital Gauge rendering
    var digitalGaugeSample = new
    ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
      // For setting text
      value: "WELCOME",
      frame: {
        // For setting inner width
        innerWidth: 6,
        // For setting outer width
        outerWidth: 10,
      },
    })
  });
}
```

Execute the above code examples to render the **DigitalGauge** as follows.

A digital gauge visualization displaying the word "WELCOME" in a pixelated, digital font.

### Setting Background Image

For a better appearance, you can set the `background image` for the **Digital Gauge** using the property `backgroundImageUrl`.

## HTML

```
<div id="DigitalGauge1"></div>
```

## JAVASCRIPT

```
$(function() {
  // For Digital Gauge rendering
  var digitalGaugeSample = new
  ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
    // For setting text
    value: "RADAR",
    height: 300,
    frame: {
      // For setting background image
      backgroundImageUrl: "board3.jpg",
    },
    items: [{
      position: {
        x: 95,
        y: 10
      }
    }
  ]
});
```

```

}
}]
})
});

```

Execute the above code examples to render the **DigitalGauge** as follows.



## Digital Elements

### Text Customization

- The attribute **value** refers the text displayed in the **Digital Gauge**. This text is applicable only for that item instead of all items. Text color is changed by using the property **textColor**.
- It is possible to align the text inside the **Digital Gauge** control by using the property **textAlign**. Two possible values for text align are as follows
  - left
  - right

### HTML

```
<div id="DigitalGauge1"></div>
```

### JAVASCRIPT

```

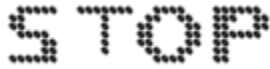
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module DigitalGaugeComponent {
$(function () {
// For Digital Gauge rendering
var digitalGaugeSample = new
ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
items: [{
// For setting alignment
textAlign: "right",

```



```
// For setting text
value: "STOP",
}]
})
});
}
```

Execute the above code examples to render the **DigitalGauge** as follows.



## Segment Settings

### Appearance

- **Digital Gauge** consists of several digital segments. **Segment** is customized with following properties.
- **Color** of the segment is set by using **color** property. Color is either given as string or hexadecimal value.
- You can add gradient effects to the segments with the help of **gradient** attribute.
- The **opacity** of the segment is also adjustable.
- The space between two segments are adjusted with **spacing** property.
- You can set the length of the text segments with **length** property.
- The width for the text segments are adjusted with **width** property.

### HTML

```
<div id="DigitalGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module DigitalGaugeComponent {
$(function () {
// For Digital Gauge rendering
var digitalGaugeSample = new
ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
width: 800,
items: [{
// For setting text
value: "GO AHEAD",
segmentSettings: {
// For setting segment color
color: "Green",
// For setting segment opacity
opacity: 0.1,
// For setting segment spacing
spacing: 4,
}
}]
})
}
```

```
});
}
```

Execute the above code examples to render the **DigitalGauge** as follows.



### Dimension Modification

- **Digital Gauge** consists of several digital segments. Segment is customized with some properties. Color of the segment is set by using **color** property. Color is either given as string or hexadecimal value.
- You can add gradient effects to the segments with the help of **gradient** attribute. The **opacity** of the segment is also adjustable. The space between two segments are adjusted with **spacing** property.

### HTML

```
<div id="DigitalGauge1"></div>
```

### JAVASCRIPT

```
$(function () {
    // For Digital Gauge rendering
    var digitalGaugeSample = new
    ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
        width: 800,
        items: [{
            // For setting text
            value: "WELCOME",
            segmentSettings: {
                // For setting segment length
                length: 3,
                // For setting segment width
                width: 3
            }
        }
    ]
    })
});
```

Execute the above code examples to render the **DigitalGauge** as follows.



## Character Settings

### Appearance

You can customize the character using **character settings**. The opacity of the character is adjustable with the help of **opacity** property. The space between two characters are adjusted with **spacing** property as like in the segment settings.

### HTML

```
<div id="DigitalGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../../tsfiles/ej.web.all.d.ts"></reference>
module DigitalGaugeComponent {
$(function () {
// For Digital Gauge rendering
var digitalGaugeSample = new
ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
width: 800,
items: [{
// For setting text
value: " Syncfusion ",
characterSettings: {
// For setting character opacity
opacity: 0.3,
// For setting character spacing
spacing: 3
}
}]
});
}
```

Execute the above code examples to render the **DigitalGauge** as follows.

Syncfusion

### Count and Type

The number of text to be displayed can be limited by the attribute called **count**. In **Digital Gauge** five different **types** of characters are supported. They are as follows,

- EightCrossEightDotMatrix
- SevenSegment
- FourteenSegment
- SixteenSegment
- EightCrossEightSquareMatrix.

### HTML

```
<div id="DigitalGauge1"></div>
```

**JAVASCRIPT**

```
$(function () {
  // For Digital Gauge rendering
  var digitalGaugeSample = new
  ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
    width: 800,
    items: [{
      // For setting text
      value: "1234567890",
      segmentSettings: {
        // For setting segment length
        length: 8,
        // For setting segment width
        width: 1
      },
      characterSettings: {
        // For setting character count
        count: 10,
        // For setting segment spacing
        spacing: 10,
        // For setting character type
        type: "sevensegment",
      }
    }]
  })
});
```

Execute the above code examples to render the **DigitalGauge** as follows.


**Text Positioning**

The text in the **DigitalGauge** is positioned with **position** object. This object contains two attributes such as **x** and **y**. The **x** variable positions the text in the horizontal axis and the **y** variable positions the text in the vertical axis.

**HTML**

```
<div id="DigitalGauge1"></div>
```

**JAVASCRIPT**

```
$(function () {
  // For Digital Gauge rendering
  var digitalGaugeSample = new
  ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
    width: 800,
    height: 300,
    frame: {
      backgroundImageUrl: "Board1.jpg"
    },
    items: [{
```

```
// For setting text
value: "YELLOW",
// For setting segment color
segmentSettings: { color: "Yellow" },
position:{
  // For setting segment x location
  x:80,
  // For setting segment y location
  y:10
}
}];
});
});
```

Execute the above code examples to render the **DigitalGauge** as follows.



### Shadow Effects

You can add the shadow effects for text using following properties.

- You can enable/disable the blurring effect for the shadows of the text using `shadow blur` property.
- You can specify the color of the text shadow using `shadow color` property.
- You can set the `x-offset` value for the shadow of the text, indicating the location where it needs to be displayed.
- You can set the `y-offset` value for the shadow of the text, indicating the location where it needs to be displayed.

### HTML

```
<div id="DigitalGauge1"></div>
```

### JAVASCRIPT

```

$(function () {
  // For Digital Gauge rendering
  var digitalGaugeSample = new
  ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
    width: 800,
    items: [{
      //For setting Text
      value: "WELCOME",
      //For setting segment length and width
      segmentSettings: {
        length: 3,
        width: 3
      },
      //For setting shadow color
      shadowColor: "yellow",
      //For setting shadow Blur
      shadowBlur: 20,
      //For setting horizontal offset
      shadowOffsetX: 15,
      //For setting vertical offset
      shadowOffsetY: 15,
    }]
  });
});

```

Execute the above code examples to render the **DigitalGauge** as follows.



### Font Customization

You can customize the font of the text as per your requirement. To customize the font, you have to set `enableCustomFont`. Following font customization options are available.

- **Font-family**- used to set the font-family of the text.
- **Font-style**- used to set the font-style of the text.
- **Font-size**- used to set the font-size of the text.

### HTML

```
<div id="DigitalCore"></div>
```

### JAVASCRIPT

```

$(function () {
  var digitalGaugeSample = new
  ej.datavisualization.DigitalGauge($("#DigitalCore"), {items:
  [{enableCustomFont: true ,font: { fontFamily: "Segou", fontStyle: "bold",
  size: "18px"}}]}]);
});

```

## Multiple Items

The text in the **Digital Gauge** is positioned with position object. This object contains two attributes such as **x** and **y**. The **x** variable positions the text in the horizontal axis and **y** variable positions the text in the vertical axis.

### HTML

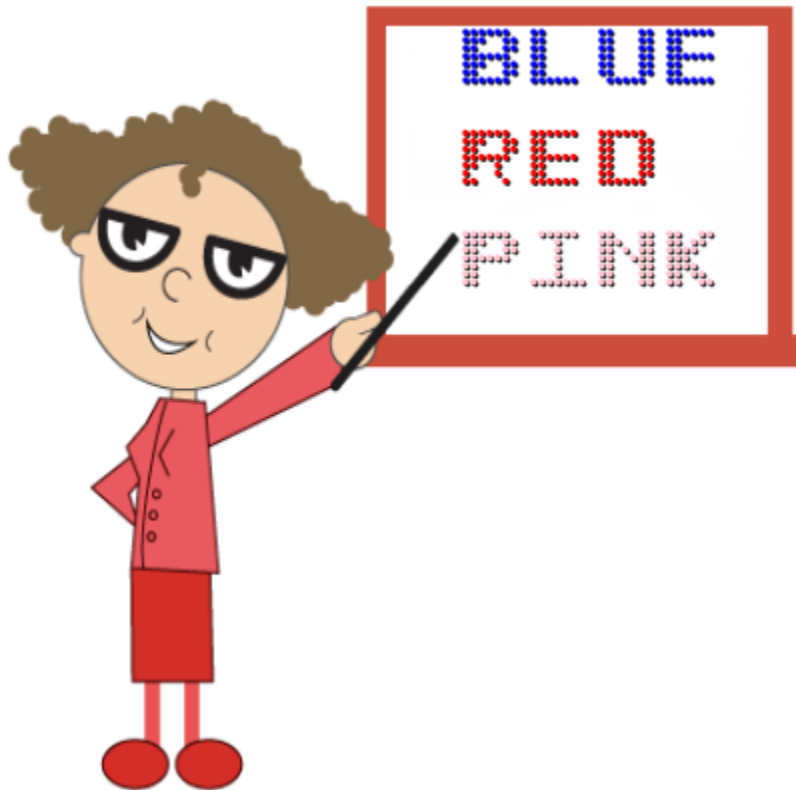
```
<div id="DigitalGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module DigitalGaugeComponent {
  $(function() {
    // For Digital Gauge rendering
    var digitalGaugeSample = new
    ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
      Width: 1350,
      height: 400,
      frame: {
        backgroundImageUrl: "Board1.png"
      },
      items: [
        // For Item 1
        {
          // For setting text
          value: "BLUE",
          segmentSettings: {
            color: "blue"
          },
          position: {
            x: 90,
            y: 0
          }
        },
        // For Item 2
        {
          // For setting text
          value: "RED",
          segmentSettings: {
            color: "red"
          },
          position: {
            x: 90,
            y: 15
          }
        },
        // For Item 3
        {
          // For setting text
          value: "PINK",
          segmentSettings: {
            color: "pink"
          },
          position: {
```

```
x: 90,  
y: 30  
}  
}  
]  
});  
});  
}
```

Execute the above code example to render the **DigitalGauge** as follows.



### Exporting the Digital Gauge

**Digital Gauge** has an exporting feature where **Gauge** control is converted into image format and then exported to client-side. The method API `exportImage` exports the **Digital Gauge**. It has two arguments such as **filename** and **file format**. For exporting, you can refer the following code example.

#### HTML

```
<div id="DigitalGauge1"></div>  
<button id="buttonSubmit">Export</button>  
<div id=" fileName ">FileName </div>  
<div id=" fileFormat ">FileFormat </div>  
<select id="fileFormat">  
<option value="JPEG">JPEG</option>  
<option value="PNG">PNG</option>  
</select>
```

#### JAVASCRIPT



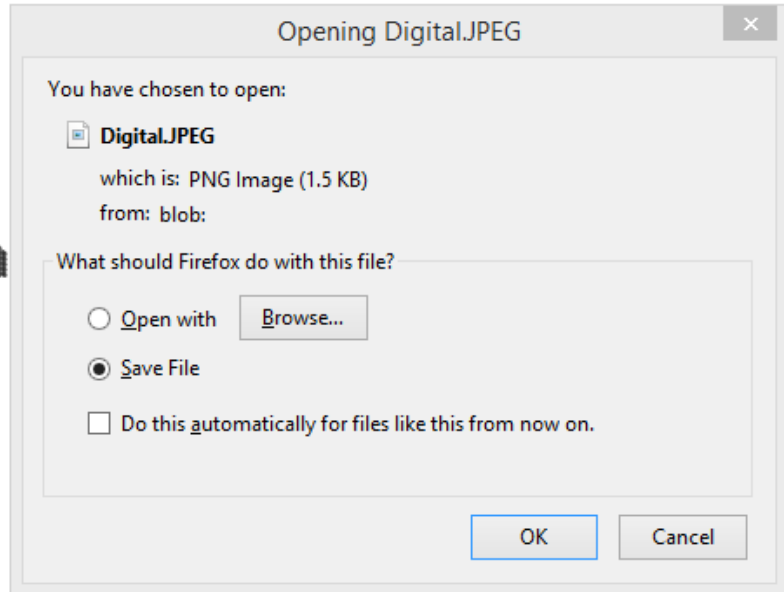
```

$(function () {
    var basicButton = new ej.Button($("#buttonSubmit"), {
        width: "50px", text: "Export", click: "buttonClickEvent",
    });
    var basicButton = new ej.DropDownList($("#fileFormat"), {
        selectedIndex: 0, width: "115px"
    });
    var digitalGaugeSample = new
    ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
        value: "Syncfusion"
    });
    });
    var digitalGaugeSample = new
    ej.datavisualization.DigitalGauge($("#DigitalGauge1"));
    digitalGaugeSample.exportImage("Digital", "JPEG");
    function buttonClickEvent() {
        var FileName = $("#fileName").val();
        var FileFormat = new ej.DropDownList($("#fileFormat"));
        FileFormat.option(value);
        var flag = new ej.datavisualization.DigitalGauge($("#DigitalGauge1"));
        flag.exportImage(FileName, FileFormat);
        if (!flag)
            alert("Sorry for the inconvenience. Export is currently not supported in
            Internet Explorer 9 and below version");
    }
}

```

Execute the above code examples to render the **DigitalGauge** as follows.

Syncfusion



## MVVM

### AngularJS

**Digital Gauge** contains AngularJS support. It is possible to add object as well as array object in the **Digital Gauge**. The two way binding support is given to the **value** for displaying the text.

### Rendering the Digital Gauge

**ej-DigitalGauge** is the control tag, where **ej** is tag prefix and **DigitalGauge** is the control name. **Digital Gauge** is rendered with the following code example.

#### HTML

```
<!--To Render the Digital gauge-->
<!doctype html>
<html ng-app="syncApp">
<head>
<!--Refer the necessary script here-->
</head>
<body ng-controller="DigitalGauge">
<ej-digitalgauge id="digitalCore" e-height="500" e-load="loadGaugeTheme">
</ej-digitalgauge>
<script type="text/javascript">
<!--binding the value to the scope variables in application controller-->
angular.module('syncApp', ['ejangular'])
.controller('DigitalGauge', function ($scope) {
$scope.number = "text";
});
</script>
</body>
</html>
```

Execute the above code to render the following output.

text

### Adding the Digital Gauge Items

**Digital Gauge** is rendered with the following code example. You can extend the Object in the array collection such as, **position**, **characterSetting**, **segmentSetting**, etc. with hyphen in the same tag.

**Example:** e-position-x.

#### HTML

```
<!--To Render the Digital gauge-->
<ej-digitalgauge id="digitalCore">
<!--Adding Item collection to the digital gauge-->
<e-items>
<e-item e-segmentSettings-width="1" e-segmentSettings-spacing="0"
e-value="Syncfusion" e-characterSetting-opacity="0.8"
e-position-x="52" e-position-y="52"></e-item>
</e-items>
</ej-digitalgauge>
```

Finally while running the above codes, the following output will be rendered.

Syncfusion

### Two Way Binding

**Digital Gauge** supports the two way binding for the property **value** as mentioned earlier. Following code example explains how to achieve the two way binding to the **Digital Gauge**.

**HTML**

```

<!doctype html>
<html ng-app="syncApp">
<head>
<meta charset="utf-8">
<!--Refer the necessary script here-->
</head>
<body ng-controller="DigitalGauge">
Type here <input type="text" id="txtValue" **ng-model="number"**
Style="width:110px"/>
<ej-digitalgauge id="digitalCore" e-height="200" e-load="loadGaugeTheme">
<e-items>
<e-item e-segmentSettings-width="1" e-segmentSettings-spacing="0"
e-characterSetting-opacity="0.8" e-position-x="52"
e-value="number" e-position-y="52"></e-item>
</e-items>
</ej-digitalgauge>
<script type="text/javascript">
<!--binding the value to the scope variables in application controller-->
angular.module('syncApp', ['ejangular'])
.controller('DigitalGauge', function ($scope) {
$scope.number = "Syncfusion";
});
</script>
</body>
</html>

```

Execute the above code to render the following output.

# Syncfusion

Methods

*destroy()*

To destroy the digital gauge

**HTML**

```
<div id="DigitalGauge1"></div>
```

**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DigitalGaugeComponent {
$(function () {
var sample = new ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
///...///
});
sample.destroy();
});
});
}

```

*exportImage(fileName, fileType)*

To export Digital Gauge as Image

### HTML

```
<div id="DigitalGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DigitalGaugeComponent {
$(function () {
var sample = new ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
///...//
});
sample.exportImage();
});
});
}
```

*getPosition(itemIndex)*

Gets the location of an item that is displayed on the gauge.

### HTML

```
<div id="DigitalGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DigitalGaugeComponent {
$(function () {
var sample = new ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
///...//
});
sample.getPosition();
});
});
}
```

*getValue(itemIndex)*

ClientSideMethod getValue Gets the value of an item that is displayed on the gauge

### HTML

```
<div id="DigitalGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DigitalGaugeComponent {
$(function () {
```

```

var sample = new ej.datavisualization.DigitalGauge($("#DigitalrGauge1"), {
    //...//
});
sample.getValue();
});
});
}

```

### *refresh()*

Refresh the digital gauge widget

### HTML

```
<div id="DigitalGauge1"></div>
```

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DigitalGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
            //...//
        });
        sample.refresh();
    });
});
}

```

### *setPosition(itemIndex, value)*

ClientSideMethod Set Position Sets the location of an item to be displayed in the gauge

### HTML

```
<div id="DigitalGauge1"></div>
```

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DigitalGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
            //...//
        });
        sample.setPosition();
    });
});
}

```

### *setValue(itemIndex, value)*

ClientSideMethod SetValue Sets the value of an item to be displayed in the gauge.

### HTML

```
<div id="DigitalGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DigitalGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.DigitalGauge($("#DigitalGauge1"), {
      ///...///
    });
    sample.setValue();
  });
}
```

### Events

#### init

Triggers when the gauge is initialized.

### JAVASCRIPT

```
<script>
  //init event for Digital gauge
  $(function () {
    var sample = new ej.datavisualization.DigitalGauge($("#gauge"), {
      init: function () {
        ///...///
      }
    });
  });
</script>
```

#### itemRendering

Triggers when the gauge item rendering.

### JAVASCRIPT

```
<script>
  //itemRendering event for Digital gauge
  $(function () {
    var sample = new ej.datavisualization.DigitalGauge($("#gauge"), {
      itemRendering: function () {
        ///...///
      }
    });
  });
</script>
```

#### load

Triggers when the gauge is start to load.

### JAVASCRIPT

```
<script>
//load event for Digital gauge
$(function () {
var sample = new ej.datavisualization.DigitalGauge($("#gauge"), {
load: function () {
//...//
}
});
});
</script>
```

### renderComplete

Triggers when the gauge render is completed.

### JAVASCRIPT

```
<script>
//renderComplete event for Digital gauge
$(function () {
var sample = new ej.datavisualization.DigitalGauge($("#gauge"), {
renderComplete: function () {
//...//
}
});
});
</script>
```

## DropDownList

### DropDownList

The DropDownList widget displays a single column list of items which enables you to select single or multiple items from the list. By default no selection is been made in the widget, the user has to navigate through the items using mouse or keyboard actions to select an item.

### Key Features

- **DataSources** - Supports various client-side and remote data sources such as JSON, RESTful services, OData services, WCF services and much more.
- **Sorting** - Sorts both ascending and descending orders.
- **Grouping** - Categorizes the list Items.
- **Searching** - Provides both incremental and Filter Search.
- **Virtual scrolling** - Fetches the data from remote on demand.
- **PopupList Resize** - Resizes the popup list in the needed dimensions.
- **Template** - Designs own layout of list items.
- **Cascading** - Cascades the data sources with multiple DropDownList.
- **State persistence** - Persistent state of properties on page refresh.
- **Responsive** - Suits responsive layouts.
- **Validation** - Built in jQuery validation.
- **Localization** - Supports localization to different cultures.
- **Selection** - Supports single or multi selection with the DropDownList and display the text in two modes, delimiter and visual mode.

- **Accessibility** - Supports keyboard and ARIA accessibility.

## Getting Started

The external script dependencies of the DropDownList widget are,

- [jQuery 1.7.1](#) and later versions.
- [jQuery.easing](#) - to support the animation effects.

And the internal script dependencies of the DropDownList widget are:

| File                   | Description / Usage   |
|------------------------|---|
| ej.core.min.js         | Must be referred always before using all the JS controls.                           |
| ej.data.min.js         | Used to handle data operation and should be used while binding data to JS controls. |
| ej.dropdownlist.min.js | The dropdownlist's main file  |
| ej.checkbox.min.js     | Should be referred when using checkbox functionalities in DropDownList.             |
| ej.scroller.min.js     | Should be referred when using scrolling in DropDownList.                            |
| ej.draggable.min.js    | Should be referred when using popup resize functionality in DropDownList.           |

For getting started you can use the 'ej.web.all.min.js' file, which encapsulates all the 'ej' controls and frameworks in one single file.

For themes, you can use the 'ej.web.all.min.css' CDN link from the snippet given. To add the themes in your application, please refer [this link](#).

## Creating DropDownList in Typescript

Create an **HTML** page and add the scripts references in the order mentioned in the following code example.

### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```



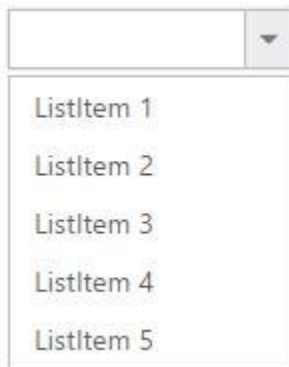
The DropDownList can be created from a HTML 'input' element with the HTML 'id' attribute and pre-defined options set to it. To create the DropDownList, you should call the 'ejDropDownList' jQuery plug-in function.

### HTML

```
<div class="control">
  <input type="text" id="dropdown1" />
</div>
```

### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  var items = [
    { text: "ListItem 1", value: "item 1" }, { text: "ListItem 2", value: "item 2" }, { text: "ListItem 3", value: "item 3" },
    { text: "ListItem 4", value: "item 4" }, { text: "ListItem 5", value: "item 5" } ];
  $(function () {
    var sample = new ej.DropDownList($("#dropdown1"), {
      dataSource: items,
      fields: {text: "text", value: "value" }
    });
  });
}
```



### Populating data

The DropDownList can be bounded to both local array and remote data services .You can bind data to DropDownList through services or local data using [dataSource](#) property

### HTML

```
<div class="control">
  <input type="text" id="dropdown1" />
</div>
```

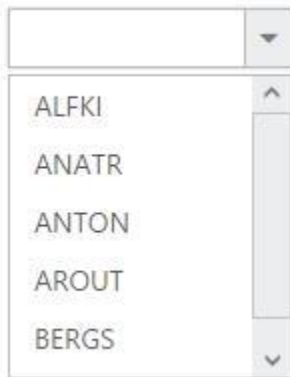
### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```

module DropDownListComponent {
$(function () {
var customers = [
{ id: "1", text: "ALFKI" }, { id: "2", text: "ANATR" }, { id: "3", text:
"ANTON" },
{ id: "4", text: "AROUT" }, { id: "5", text: "BERGS" }, { id: "6", text:
"BLAUS" }
];
var sample = new ej.DropDownList($("#dropdown1"), {
dataSource: new ej.DataManager(customers),
fields: { id: "id", text: "text", value: "text" }
});
})
}

```



### Setting Dimensions

DropDownList dimensions can be set using width and height API.

#### HTML

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
var items = [
{ text: "ListItem 1", value: "item 1" }, { text: "ListItem 2", value: "item
2" }, { text: "ListItem 3", value: "item 3"},
{ text: "ListItem 4", value: "item 4" }, { text: "ListItem 5", value: "item
5" }];
$(function () {
var sample = new ej.DropDownList($("#dropdown1"), {
dataSource: items,
fields: { text: "text", value: "value" },
width: "200px",
height: "50px"
});
});
}

```

### Setting dimensions to Popup list

PopupWidth and popupHeight can be used to create a fixed size popup list.

#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
var items = [
{ text: "ListItem 1", value: "item 1" }, { text: "ListItem 2", value: "item 2" }, { text: "ListItem 3", value: "item 3"},
{ text: "ListItem 4", value: "item 4" }, { text: "ListItem 5", value: "item 5" }];
$(function () {
var sample = new ej.DropDownList($("#dropdown1"), {
dataSource: items,
fields: { text: "text", value: "value" },
width: "200px",
height: "50px",
popupHeight: "100px",
popupWidth: "200px"
});
});
}
```

### Setting and Getting Value

You can select single or multiple values from DropDownList widget. To assign a value initially to the DropDownList, you can use [value](#) property.

---

**Note:** To select multiple items based on index, refer [here](#).

---

### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
var items = [
{ text: "ListItem 1", value: "item 1" }, { text: "ListItem 2", value: "item 2" }, { text: "ListItem 3", value: "item 3"},
{ text: "ListItem 4", value: "item 4" }, { text: "ListItem 5", value: "item 5" }];
$(function () {
var sample = new ej.DropDownList($("#dropdown1"), {
dataSource: items,
fields: { text: "text", value: "value" },
value: "item 3",
change: function ()
{
var obj = $('#dropdown1').data("ejDropDownList");
console.log("Selected Item's Text - " + obj.option("text"));
console.log("selected Item's Value - " + obj.option("value"));
}
});
});
}
```

### Rendering Mode

DropDownList widget can be created in three ways.

- Using an input element
- Using a select element
- Using UL-LI

### Using an input element

Create an input element with the HTML "id" attribute set to it. To initialize the DropDownList, you should call the "ejDropDownList" jQuery plug-in function with the options as parameter

You can bind the local JSON array data source to the DropDownList using [dataSource](#) and [fields](#) properties. Fields property is used to map with the corresponding columns.

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  var items = [{ text: "ListItem 1", value: "item1" },
    { text: "ListItem 2", value: "item2" },
    { text: "ListItem 3", value: "item3" },
    { text: "ListItem 4", value: "item4" },
    { text: "ListItem 5", value: "item5" }];
  $(function () {
    var sample = new ej.DropDownList($("#dropdown1"), {
      dataSource: items,
      fields: { text: "text", value: "value" }
    });
  });
}
```

### Using Select Element

You can create a DropDownList using a select element and the detailed information is given in [creating DropDownList](#) section.

### Using UL-LI

You can bind the predefined set of UL-LI elements to generate the list of popup items. These items can be customized by adding any images, div elements, radio buttons, text boxes etc.

Create a div with UL-LI elements and assign that div id into [targetID](#) property and initialize the widget.

### HTML

```
<input type="text" id="dropdown1" />
<div id="tools">
  <ul>
    <li>
      <div class="mailtools categorize"></div>
      Categorize and Move</li>
    <li>
      <div class="mailtools done"></div>
      Done</li>
    <li>
```

```
<div class="mailtools flag"></div>
Flag & Move</li>
<li>
<div class="mailtools forward"></div>
Forward</li>
<li>
<div class="mailtools movetofolder"></div>
Move to Folder</li>
<li>
<div class="mailtools newmail"></div>
New E-mail</li>
<li>
<div class="mailtools meeting"></div>
New Meeting</li>
<li>
<div class="mailtools reply"></div>
Reply & Delete</li>
</ul>
</div>
```

## CSS

```
.mailtools {
display: block;
background-image: url('iconsapps.png');
height: 25px;
width: 25px;
background-position: center center;
background-repeat: no-repeat;
}
.mailtools.done {
background-position: 0 0;
}
.mailtools.movetofolder {
background-position: 0 -22px;
}
.mailtools.categorize {
background-position: 0 -46px;
}
.mailtools.flag {
background-position: 0 -70px;
}
.mailtools.forward {
background-position: 0 -94px;
}
.mailtools.newmail {
background-position: 0 -116px;
}
.mailtools.reply {
background-position: 0 -140px;
}
.mailtools.meeting {
background-position: 0 -164px;
}
```

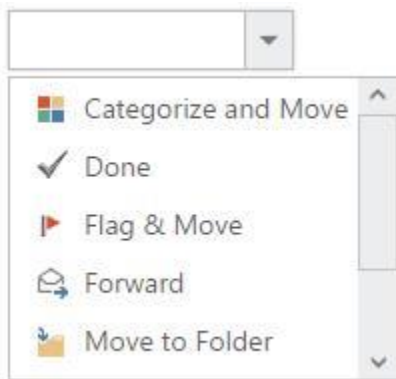
**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
$(function() {
var sample = new ej.DropDownList($("#dropdown1"), {
targetID: "tools"
});
});
}

```

**Note:** Images for this sample are available in (installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\samples\web\themes\images<br/>



**Note:** Any EJ widget can be embedded in the target element but the default action of that widget should be prevented in order to create custom actions. To show a sample, integration with [TreeView](#) widget is demonstrated [here](#).

**Data Binding**

To populate data in the DropDownList widget, define [dataSource](#) property with associated fields. In DropDownList, can bind either local array or OData, WebApi and other [RESTful](#) services.

**Fields**

The below listed fields are the data collection fields which maps fields for the data items of the DropDownList.

| Properties | Description  |
|------------|--|
| dataSource | The data source contains the list of data for generating the popup list's items. |
| query      | It specifies the query to retrieve the data from the online server.              |
| fields     | It specifies the mapping fields for the data items of the DropDownList widget.   |
| id         | It specifies the ID of the tag.  |
| text       | It specifies the text content of the tag.  |
| value      | It specifies the value of the tag.   |
| groupBy    | It is used to categorize the items based on a specific field.                    |

|                 |  |
|-----------------|--|
| imageUrl        | It defines the image location.   |
| imageAttributes | It defines the image attributes such as height, width, styles, etc.  |
| spriteCssClass  | It defines the sprite CSS for the image tag.   |
| htmlAttributes  | It defines the HTML attributes such as class and styles for an item.   |
| selected        | This field defines the tag value to be selected initially. Corresponding field mapped has Boolean values to select the list items on control creation. The data with value "true" in this field is selected automatically when the control is initialized with checkbox. |
| tableName       | It defines the table name for the tag value or displays text while rendering remote data.  |

### Local Data

Define a JSON array and initialize the widget with [dataSource](#) property. Specify the column names in the [fields](#) property.

**Note:** The columns are bounded automatically when the fields are specified with the default names like id, text, etc...

### HTML

```
<input type="text" id="dropdown1" />
```

### CSS

```
.imgId {
margin: 0;
padding: 3px 10px 3px 3px;
border: 0 none;
width: 60px;
height: 60px;
float: none;
}
```

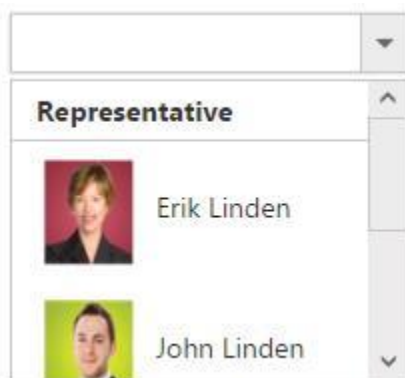
### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
var List = [{
text: "Erik Linden",
role: "Representative",
country: "England",
img: "images/Employee/3.png",
attr: {
class: "imgId"
}
}, {
text: "John Linden",
role: "Representative",
```

```

country: "Norway",
img: "images/Employee/6.png",
attr: {
  class: "imgId"
}, {
text: "Louis",
role: "Representative",
country: "Australia",
img: "images/Employee/7.png",
attr: {
  class: "imgId"
}, {
text: "Lawrence",
role: "Representative",
country: "India",
img: "images/Employee/8.png",
attr: {
  class: "imgId"
}
}];
$(function () {
  var sample = new ej.DropDownList($("#dropdown1"), {
    dataSource: List,
    fields: {
      text: "text",
      value: "country",
      groupBy: "role",
      imageUrl: "img",
      imageAttributes: "attr"
    },
    width: "200px"
  });
});
}

```



**Note:** Images for this sample are available in (installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\samples\web\themes\images<br/>

**Information:** htmlAttributes and imageAttributes should have JSON type value and sample for spriteCSSClass field is available in [here](#)



The JSON array to the [dataSource](#) property can also be provided as an instance of the [ej.DataManager](#). When the JSON array is passed as an instance of [ej.DataManager](#), the [ej.JsonAdaptor](#) will be used to manipulate the DropDownList data source. The following code explains this behavior,

#### HTML

```
<input type="text" id="dropdown1" />
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  var items = [{
    text: "ListItem 1",
    value: "item1"
  }, {
    text: "ListItem 2",
    value: "item2"
  }, {
    text: "ListItem 3",
    value: "item3"
  }, {
    text: "ListItem 4",
    value: "item4"
  }, {
    text: "ListItem 5",
    value: "item5"
  }];
  //Passing as instance to the DataManager
  $(function () {
    var sample = new ej.DropDownList($("#dropdown1"), {
      dataSource: ej.DataManager(items)
    });
  });
}
```

#### Binding Remote Data Service

To bind remote data to the DropDownList, assign a service data as an instance of [ejDataManager](#) to the [dataSource](#) property.

#### OData

OData is a standardized protocol for creating and consuming data. Provide the [OData service](#) URL directly to the "ej.DataManager" class and then you can assign it to DropDownList "dataSource".

#### HTML

```
<input type="text" id="dropdown1" />
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  $(function () {
```

```
var dataManager =  
ej.DataManager("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders");  
var sample = new ej.DropDownList($("#dropdown1"), {  
  dataSource: dataManager,  
  fields: {  
    text: "ShipCountry",  
    value: "OrderID"  
  }  
});  
});  
});
```

### Virtual Scrolling

To improve the performance when displaying large data set, you can use “allowVirtualScrolling” and [virtualScrollMode](#) property. This retrieves only a fixed amount of list items and loads remaining data on scrolling. The items will be fetched via AJAX request.

This supports two modes of virtualization. They are,

- Normal Mode
- Continuous Mode

---

**Information:** 1. Sorting and Grouping is not supported with Virtual Scrolling

---

**Information:** 2. “virtualScrollMode” property accepts both the string and ej.VirtualScrollMode enum value.

---

### Normal Mode

It loads the data on scrolling the list of items. This can be achieved by setting [normal](#) value to the "virtualScrollMode" property.

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module DropDownListComponent {  
  $(function() {  
    var dataManager = ej.DataManager({  
      url: "http://mvc.syncfusion.com/services/Northwnd.svc/Orders"  
    });  
    var sample = new ej.DropDownList($("#dropdown1"), {  
      dataSource: dataManager,  
      fields: {  
        text: "ShipName",  
        value: "ShipCountry"  
      },  
      allowVirtualScrolling: true,  
      virtualScrollMode: ej.VirtualScrollMode.Normal,  
      itemCount: 7  
    });  
  });  
}
```

```
}};
}
```

### Continuous Mode

It loads the set of items when the scroller reaches at the end. This behaves like infinity scrolling. So when scroll reaches the end, it will fetch the remaining set of items and bind with your DropDownList. This can be achieved by setting [continuous](#) value to the "virtualScrollMode" property.

**Note:** In both modes, set of items will be fetched based on the count specified in the [itemsCount](#) property and next set of items will be loaded on scrolling.

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  $(function() {
    var dataManager = ej.DataManager({
      url: "http://mvc.syncfusion.com/services/Northwnd.svc/Orders"
    });
    var sample = new ej.DropDownList($("#dropdown1"), {
      dataSource: dataManager,
      fields: {
        text: "ShipName",
        value: "ShipCountry"
      },
      allowVirtualScrolling: true,
      virtualScrollMode: ej.VirtualScrollMode.Continuous,
      itemsCount: 7
    });
  });
}
```

### Checkbox

DropDownList displays checkboxes to the left of each item when you set [showCheckbox](#) property to true. It allows you to select more than one item at a time from DropDownList. Popup list stays open until the user finishes selection. When you click on an item's text or checkbox then the checkbox checked status get change.

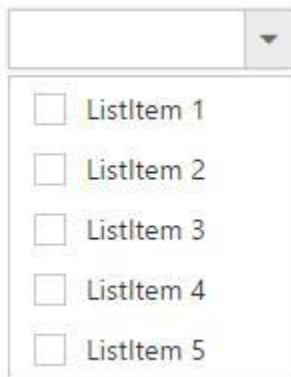
### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  var items = [{
    text: "ListItem 1",
```

```
value: "item1"
}, {
text: "ListItem 2",
value: "item2"
}, {
text: "ListItem 3",
value: "item3"
}, {
text: "ListItem 4",
value: "item4"
}, {
text: "ListItem 5",
value: "item5"
}];
$(function () {
var sample = new ej.DropDownList($("#dropdown1"), {
dataSource: items,
fields: {
text: "text",
value: "value"
},
showCheckbox: true
});
});
}
```



**Note:** if you want to showcase the DropDownList with default checked items on data binding, specify selected field with Boolean values.

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
var items = [{
text: "ListItem 1",
value: "item1",
selected: true
}
```

```
}, {
  text: "ListItem 2",
  value: "item2",
  selected: false
}, {
  text: "ListItem 3",
  value: "item3",
  selected: true
}, {
  text: "ListItem 4",
  value: "item4",
  selected: false
}, {
  text: "ListItem 5",
  value: "item5",
  selected: false
}];
$(function () {
  var sample = new ej.DropDownList($("#dropdown1"), {
    width: 300,
    dataSource: items,
    fields: {
      text: "text",
      value: "value",
      selected: "selected"
    },
    showCheckbox: true
  });
});
});
}
```



### Selection Modes

The [multiSelectMode](#) property enables you to make multiple selections in the following two ways:

- Delimiter
- Visual Mode

---

**Information:** “multiSelectMode” property accepts both the **string** and **ej.MultiSelectMode** enum value.

---

### Delimiter

Each checked item's text is appended to the textbox with delimiter “,” by default. This is enabled by assigning “**delimiter**” (string) or **ej.MultiSelectMode.Delimiter** (enum) value to multiSelectMode property. You can customize the delimiter option by using [delimiterChar](#) property.

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
    var items = [{
        text: "ListItem 1",
        value: "item1"
    }, {
        text: "ListItem 2",
        value: "item2"
    }, {
        text: "ListItem 3",
        value: "item3"
    }, {
        text: "ListItem 4",
        value: "item4"
    }, {
        text: "ListItem 5",
        value: "item5"
    }
    ];
    $(function () {
        var sample = new ej.DropDownList($("#dropdown1"), {
            width: 300,
            dataSource: items,
            fields: {
                text: "text",
                value: "value"
            },
            showCheckbox: true,
            multiSelectMode: ej.MultiSelectMode.Delimiter,
            delimiterChar: "-"
        });
    });
}
```



### Visual Mode

When you enable this option in DropDownList widget, each checked item's text is appended to the text box in a box model layout. This is enabled by assigning **"visualmode"** (string) or **ej.MultiSelectMode.VisualMode** (enum) value to multiSelectMode property.

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
    var items = [{
        text: "ListItem 1",
        value: "item1"
    }, {
        text: "ListItem 2",
        value: "item2"
    }, {
        text: "ListItem 3",
        value: "item3"
    }, {
        text: "ListItem 4",
        value: "item4"
    }, {
        text: "ListItem 5",
        value: "item5"
    }];
    $(function () {
        var sample = new ej.DropDownList($("#dropdown1"), {
            width: 300,
            dataSource: items,
            fields: {
                text: "text",
                value: "value"
            },
            showCheckbox: true,
            multiSelectMode: ej.MultiSelectMode.VisualMode
        });
    });
}
```



### Check/Uncheck All

You can check/uncheck all the list items at run time by using [checkAll](#) and [uncheckAll](#) method. By default no item will be in checked state.

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
$(function () {  
    var sample = new ej.DropDownList($("#dropdown1"), {  
        var items = [{  
            text: "ListItem 1",  
            value: "item1"  
        }, {  
            text: "ListItem 2",  
            value: "item2"  
        }, {  
            text: "ListItem 3",  
            value: "item3"  
        }, {  
            text: "ListItem 4",  
            value: "item4"  
        }, {  
            text: "ListItem 5",  
            value: "item5"  
        }  
    ];  
    $(function () {  
        var sample = new ej.DropDownList($("#dropdown1"), {  
            dataSource: items,  
            fields: {  
                text: "text",  
                value: "value"  
            },  
            showCheckbox: true  
        });  
        var target = new ej.ToggleButton($("#toggle"), {  
            "change": "onCheckUncheckAll",  
            "defaultText": "Check All",  
            "activeText": "Uncheck All"  
        });  
    });  
});
```



```

});
});
}
function onCheckUncheckAll(args) {
if (args.isChecked) target.checkAll();
else target.uncheckAll();
}

```

## Functionalities

### Selection

By default only one item can be selected from the popup list. For multiple selection, you have to enable [checkboxes](#). The selected item consist of active class ("e-active") to differentiate it from other items.

The following API's, select the items in the DropDownList via text or value or indices.

| Properties                      | Description  |
|---------------------------------|--|
| <a href="#">value</a>           | To select an item initially, you can pass the item's value via value property.<br>Multiple items can select via value property, the given values should be separated by delimiter character. |
| <a href="#">text</a>            | To select an item initially, you can pass the item's text via text property.<br>Multiple items can select via text property, the given values should be separated by delimiter character.    |
| <a href="#">selectedIndex</a>   | Select a single item by passing an index value to the selectedIndex property.  |
| <a href="#">selectedIndices</a> | Select more than one items by passing index values to the selectedIndices property when multi selection enabled.   |

**Note:** Index starts from 0 here.

To use "selectedIndices" property, you should enable with showCheckbox or multiSelectMode property.

The following methods, select the items in the DropDownList.

| Methods                             | Description   |
|-------------------------------------|---|
| <a href="#">selectItemByIndices</a> | This method is used to select the list of items in the DropDownList through the Index of the items. |
| <a href="#">selectItemByText</a>    | This method is used to select an item in the DropDownList by using the given text value.            |
| <a href="#">selectItemByValue</a>   | This method is used to select an item in the DropDownList by using the given value.                 |

The following methods, used to retrieve the items from the DropDownList.

| Methods                          | Description  |
|----------------------------------|--|
| <a href="#">getListData</a>      | This method is used to retrieve the items that are bound with the DropDownList.        |
| <a href="#">getSelectedItem</a>  | This method is used to get the selected items in the DropDownList.                     |
| <a href="#">getSelectedValue</a> | This method is used to retrieve the items value that are selected in the DropDownList. |

**Information:** When multiSelectMode is enabled in a DropDownList and selected items having same text but its value is different means, the items can be selected. Please refer the online [link](#)

*Using value or text*

To select an item initially you can pass the item's value via [value](#) property or [selectItemByValue](#) method. To achieve this DropDownList widget must be initiated with the associate value.

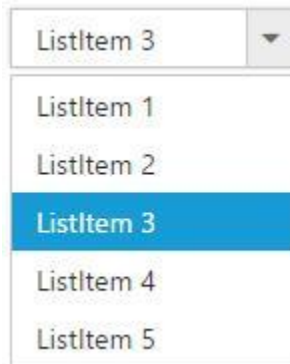
### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  var items = [{
    text: "ListItem 1",
    value: "item1"
  }, {
    text: "ListItem 2",
    value: "item2"
  }, {
    text: "ListItem 3",
    value: "item3"
  }, {
    text: "ListItem 4",
    value: "item4"
  }, {
    text: "ListItem 5",
    value: "item5"
  }];
  $(function() {
    var sample = new ej.DropDownList($("#dropdown1"), {
      dataSource: items,
      fields: {
        text: "text",
        value: "value"
      },
      value: "item3"
    });
  });
}
```

```
});  
});  
}
```



**Note:** To retrieve the selected item's LI elements and value you can use [getSelectedItem](#), [getSelectedValue](#) methods respectively.

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module DropDownListComponent {  
  var items = [{  
    text: "ListItem 1",  
    value: "item1"  
  }, {  
    text: "ListItem 2",  
    value: "item2"  
  }, {  
    text: "ListItem 3",  
    value: "item3"  
  }, {  
    text: "ListItem 4",  
    value: "item4"  
  }, {  
    text: "ListItem 5",  
    value: "item5"  
  }];  
  $(function() {  
    var sample = new ej.DropDownList($("#dropdown1"), {  
      dataSource: items,  
      fields: {  
        text: "text",  
        value: "value"  
      },  
      selectedIndex: 0  
    });  
    var obj = $('#dropdown1').data("ejDropDownList");  
    //the below given code will return the li element of the selected item
```

```
console.log(obj.getSelectedItem());  
//the below given code will return the JSON object of the selected item  
console.log(JSON.parse(obj.getSelectedValue()));  
});  
}
```

### Using indices

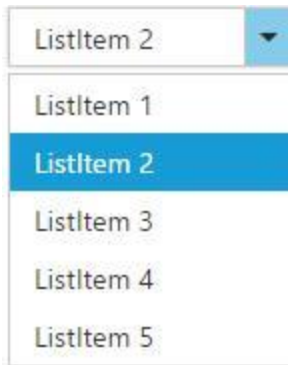
You can select a single or more than one item by passing index values to the properties [selectedIndex](#) or [selectedIndices](#) respectively. Index starts from 0 here.

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module DropDownListComponent {  
  var items = [{  
    text: "ListItem 1",  
    value: "item1"  
  }, {  
    text: "ListItem 2",  
    value: "item2"  
  }, {  
    text: "ListItem 3",  
    value: "item3"  
  }, {  
    text: "ListItem 4",  
    value: "item4"  
  }, {  
    text: "ListItem 5",  
    value: "item5"  
  }];  
  $(function() {  
    var sample = new ej.DropDownList($("#dropdown1"), {  
      dataSource: items,  
      fields: {  
        text: "text",  
        value: "value"  
      },  
      selectedIndex: 1  
    });  
  });  
}
```



**Information:** To use "selectedIndices" property, you should enable either showCheckbox or multiSelectMode property First.

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
    var items = [{
        text: "ListItem 1",
        value: "item1"
    }, {
        text: "ListItem 2",
        value: "item2"
    }, {
        text: "ListItem 3",
        value: "item3"
    }, {
        text: "ListItem 4",
        value: "item4"
    }, {
        text: "ListItem 5",
        value: "item5"
    }];
    $(function() {
        var sample = new ej.DropDownList($("#dropdown1"), {
            dataSource: items,
            fields: {
                text: "text",
                value: "value"
            },
            showCheckbox: true,
            selectedIndices: [1, 2]
        });
    });
}
```

### Unselect items

Similarly, you can unselect a single or multiple items by using [unselectItemByValue](#) or [unselectItemByIndices](#) or [unselectItemByText](#) methods. This will remove the selection state of the corresponding data item from the popup list and textbox.

### HTML

```
<input type="text" id="dropdown1" />
<input type="button" value="Unselect" onclick="unselect()" />
```

### JAVASCRIPT

```
var obj;
$(function() {
var items = [{
text: "ListItem 1",
value: "item1"
}, {
text: "ListItem 2",
value: "item2"
}, {
text: "ListItem 3",
value: "item3"
}, {
text: "ListItem 4",
value: "item4"
}, {
text: "ListItem 5",
value: "item5"
}];
$('#dropdown1').ejDropDownList({
width: 300,
dataSource: items,
fields: {
text: "text",
value: "value"
},
showCheckbox: true,
selectedIndices: [1, 2, 3]
});
obj = $('#dropdown1').data("ejDropDownList");
console.log("Selected Item's Text - " + obj.option("text"));
console.log("selected Item's Value - " + obj.option("value"));
});
function unselect() {
obj.unselectItemByValue("item2");
obj.unselectItemByIndices(2);
obj.unselectItemByText("ListItem 4");
}
```

### Grouping

The DropDownList items can be categorized by using a specific field in the popup list. This is enabled by using [groupBy](#) field on data source binding. By default grouping is disabled in DropDownList.

The below given example explains the behavior of grouping with JSON array binding.

## HTML

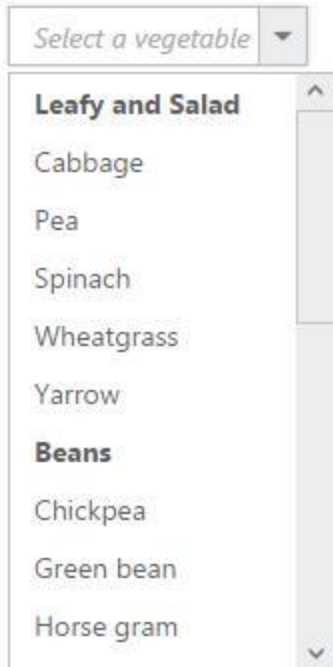
```
<input type="text" id="dropdown1" />
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  var skills = [{
    skill: "Cabbage",
    category: "Leafy and Salad"
  }, {
    skill: "Pea",
    category: "Leafy and Salad"
  }, {
    skill: "Spinach",
    category: "Leafy and Salad"
  }, {
    skill: "Wheat grass",
    category: "Leafy and Salad"
  }, {
    skill: "Yarrow",
    category: "Leafy and Salad"
  }, {
    skill: "Chickpea",
    category: "Beans"
  }, {
    skill: "Green bean",
    category: "Beans"
  }, {
    skill: "Horse gram",
    category: "Beans"
  }, {
    skill: "Peanut",
    category: "Beans"
  }, {
    skill: "Pigeon pea",
    category: "Beans"
  }, {
    skill: "Garlic",
    category: "Bulb and Stem"
  }, {
    skill: "Garlic Chives",
    category: "Bulb and Stem"
  }, {
    skill: "Lotus root",
    category: "Bulb and Stem"
  }, {
    skill: "Nopal",
    category: "Bulb and Stem"
  }, {
    skill: "Onion",
    category: "Bulb and Stem"
  }, {
    skill: "Shallot",
```

```
category: "Bulb and Stem"
}, {
skill: "Beetroot",
category: "Root and Tuberous"
}, {
skill: "Carrot",
category: "Root and Tuberous"
}, {
skill: "Ginger",
category: "Root and Tuberous"
}, {
skill: "Potato",
category: "Root and Tuberous"
}, {
skill: "Radish",
category: "Root and Tuberous"
}, {
skill: "Turmeric",
category: "Root and Tuberous"
}];
$(function () {
var sample = new ej.DropDownList($("#dropdown1"), {
width: 150,
popupHeight: 300,
watermarkText: "Select a vegetable",
dataSource: skills,
fields: {
text: "skill",
groupBy: "category"
}
});
});
}
```





**Note:** Grouping has restrictions in the following scenarios,<BR>

1. It is not supported on using HTML "select" element with predefined set of options
2. When using UL-LI elements you need to use "e-category" class in LI element to specify it as the grouping header. The following code will explain this behavior,
3. The sorting behavior varies when grouping is enabled in the DropDownList, based on browser as we have used browser based stable sorting method when there is multiple level of sorting.
4. To overcome this behavior on sorting order with browser, we suggest you to set ej.support.stableSort as false from the script when the page is loaded or in document ready function.

#### JAVASCRIPT

```
$(document).ready(function () {  
    ej.support.stableSort = false;  
});
```

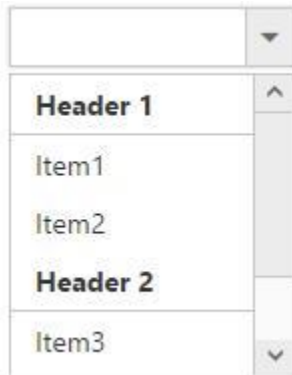
#### HTML

```
<input type="text" id="dropdown1" />
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module DropDownListComponent {  
    $(function() {  
        var sample = new ej.DropDownList($("#dropdown1"), {  
            targetID: "dropDownItems"  
        });  
    });  
}
```

```
});
}
```



**Information:** Virtual scrolling is not supported with Grouping.

### Sorting

Sorting is enabled to order to display the items alphabetically in either ascending or descending order. By default the items is displayed in the initialized order, use [enableSorting](#) property to automatically sort strings based on text field value. You can assign either “ascending” or “descending” string values to the [sortOrder](#) property to sort out the list items. By default ascending order is followed when "sortOrder" property is not specified.

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  var items = [{
    text: "ListItem 1",
    value: "item1"
  }, {
    text: "ListItem 5",
    value: "item5"
  }, {
    text: "ListItem 4",
    value: "item4"
  }, {
    text: "ListItem 2",
    value: "item2"
  }, {
    text: "ListItem 3",
    value: "item3"
  }, ];
  $(function() {
    var sample = new ej.DropDownList($("#dropdown1"), {
      dataSource: items,
      fields: {
        text: "text",
```

```
value: "value"
},
enableSorting: true,
sortOrder: "ascending"
});
});
}
```

---

**Information:** Virtual scrolling is not supported with Sorting.

---

### Cascading

This works for series of DropDownList in which items are filtered based on the previous DropDownList's selection. Cascading is performed based on the value field and this field should be bounded with a foreign key. To perform cascading, specify the child DropDownList's id in [cascadeTo](#) property and use delimiter (" , ") to specify more than one child DropDownList.

Configuring the data items for cascading to the series of DropDownList is demonstrated below

### HTML

```
<div style="width: 400px;">
<div style="float: left;">
<span>Select Group</span>
<input id="groups" type="text" />
</div>
<div style="float: right;">
<span>Select Country</span>
<input id="countries" type="text" />
</div>
</div>
```

### JAVASCRIPT

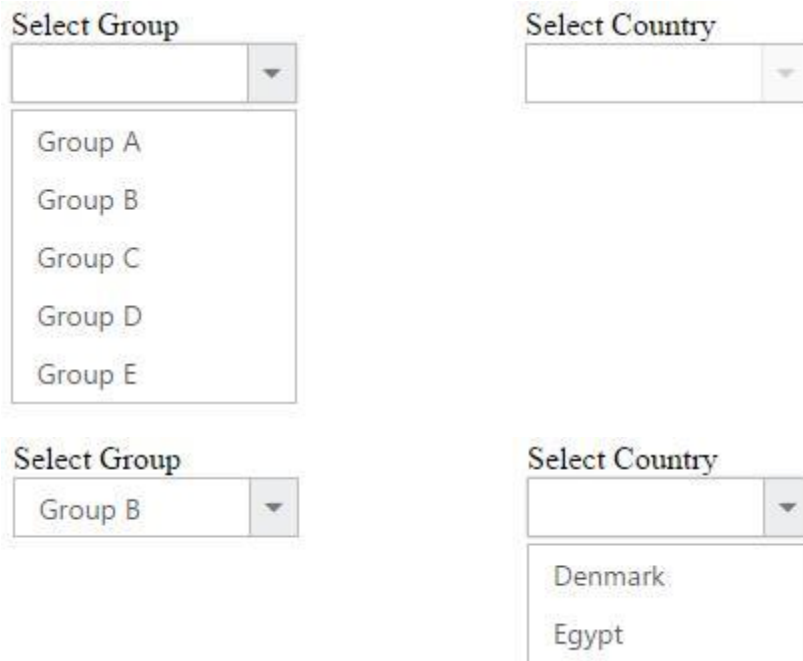
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  //first dropdown
  var groups = [{
    parentId: 'a',
    text: "Group A"
  }, {
    parentId: 'b',
    text: "Group B"
  }, {
    parentId: 'c',
    text: "Group C"
  }, {
    parentId: 'd',
    text: "Group D"
  }, {
    parentId: 'e',
    text: "Group E"
  }];
  //second dropdown
  var countries = [{
```

```
value: 11,
parentId: 'a',
text: "Algeria"
}, {
value: 12,
parentId: 'a',
text: "Armenia"
}, {
value: 13,
parentId: 'a',
text: "Bangladesh"
}, {
value: 14,
parentId: 'a',
text: "Cuba"
}, {
value: 15,
parentId: 'b',
text: "Denmark"
}, {
value: 16,
parentId: 'b',
text: "Egypt"
}, {
value: 17,
parentId: 'c',
text: "Finland"
}, {
value: 18,
parentId: 'c',
text: "India"
}, {
value: 19,
parentId: 'c',
text: "Malaysia"
}, {
value: 20,
parentId: 'd',
text: "New Zealand"
}, {
value: 21,
parentId: 'd',
text: "Norway"
}, {
value: 22,
parentId: 'd',
text: "Poland"
}, {
value: 23,
parentId: 'e',
text: "Romania"
}, {
value: 24,
parentId: 'e',
text: "Singapore"
}, {
value: 25,
```

```

parentId: 'e',
text: "Thailand"
}, {
value: 26,
parentId: 'e',
text: "Ukraine"
}];
$(function() {
var sample = new ej.DropDownList($("#groups"), {
dataSource: groups,
fields: {
text: "text",
value: "parentId"
},
cascadeTo: 'countries'
});
var sample = new ej.DropDownList($("#countries"), {
dataSource: countries,
enabled: false
});
});
}

```



#### *Binding the data source to the cascading DropDownList using cascade event*

Bind the data source to the cascading DropDownList dynamically using [cascade](#) event as demonstrated below,

#### **HTML**

```

<div style="width: 530px;">
<div style="float: left;">
<span>Select Group</span>
<input id="groups" type="text" />

```

```

</div>
<div style="float: left; padding-left: 50px;">
<span>Select Country</span>
<input id="countries" type="text" />
</div>
<div style="float: right;">
<span>Select Players</span>
<input id="players" type="text" />
</div>
</div>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  //first dropdown
  var groups = [{
    parentId: 'a',
    text: "Group A"
  }, {
    parentId: 'b',
    text: "Group B"
  }];
  var sample = new ej.DropDownList($("#groups"), {
    dataSource: groups,
    fields: {
      text: "text",
      value: "parentId"
    },
    cascadeTo: "countries,players",
    cascade: 'onChange'
  });
  var sample1 = new ej.DropDownList($("#countries"), {
    enabled: false
  });
  var sample2 = new ej.DropDownList($("#players"), {
    enabled: false
  });
  function onChange(args) {
    var countries = [{
      parentId: 'a',
      text: "Algeria"
    }, {
      parentId: 'a',
      text: "Armenia"
    }, {
      parentId: 'a',
      text: "Bangladesh"
    }, {
      parentId: 'a',
      text: "Cuba"
    }, {
      parentId: 'b',
      text: "Denmark"
    }
  ]
  }
}

```

```

}, {
  parentId: 'b',
  text: "Egypt"
}];
var players = [{
  parentId: 'a',
  text: "Adams"
}, {
  parentId: 'a',
  text: "Clarke"
}, {
  parentId: 'b',
  text: "Brett"
}, {
  parentId: 'b',
  text: "James"
}];
sample1.option({
  "dataSource": countries,
  "enabled": true
});
sample2.option({
  "dataSource": players,
  "enabled": true
});
}

```



### Multi-Level Cascading

The below scenario can be explained with three DropDownList for the multi-level cascading

### HTML

```

<div class="frame">
  <div class="control" style="float: left; padding:10px;">
    <span class="txt">Select Continent</span>
    <input id="groups" type="text" />
  </div>
  <div class="control" style="float: left; padding:10px;">
    <span class="txt">Select Country</span>
    <input id="countries" type="text" />
  </div>
  <div class="control" style="float: left; padding:10px;">
    <span class="txt">Select State</span>
    <input id="capitalList" type="text" />
  </div>
</div>

```

**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
    // declaration
    var groups = [
        { parentId: 'a', text: "Africa" },
        { parentId: 'b', text: "Asia" },
        { parentId: 'c', text: "Europe" },
        { parentId: 'd', text: "North America" },
        { parentId: 'e', text: "South America" },
        { parentId: 'f', text: "Oceania" },
        { parentId: 'g', text: "Antarctica" }]
    //Countries List
    var countries = [
        { value: 11, parentId: 'a', text: "Algeria" },
        { value: 12, parentId: 'a', text: "Egypt" },
        { value: 13, parentId: 'b', text: "Armenia" },
        { value: 14, parentId: 'b', text: "Bangladesh" },
        { value: 15, parentId: 'b', text: "India" },
        { value: 16, parentId: 'c', text: "Denmark" },
        { value: 17, parentId: 'c', text: "Finland" },
        { value: 18, parentId: 'd', text: "Cuba" },
        { value: 19, parentId: 'd', text: "USA" },
        { value: 20, parentId: 'e', text: "Brazil" },
        { value: 21, parentId: 'e', text: "Peru" },
        { value: 22, parentId: 'f', text: "Australia" },
        { value: 23, parentId: 'f', text: "New Zealand" },
        { value: 24, parentId: 'g', text: "French Southern" },
        { value: 25, parentId: 'g', text: "South Georgia" }]
    //Capital List
    var capital = [
        { value: 11, text: "Algiers" },
        { value: 12, text: "Cairo" },
        { value: 13, text: "Yerevan" },
        { value: 14, text: "Dhaka" },
        { value: 15, text: "New Delhi" },
        { value: 16, text: "Copenhagen" },
        { value: 17, text: "Helsinki" },
        { value: 18, text: "Havana" },
        { value: 19, text: "Washington, D.C." },
        { value: 20, text: "Brasilia" },
        { value: 21, text: "Lima" },
        { value: 22, text: "Canberra" },
        { value: 23, text: "Wellington" },
        { value: 24, text: "Alfred Faure" },
        { value: 25, text: "King Edward Point" }]
    $(function() {
        var sample1 = new ej.DropDownList($("#groups"), {
            dataSource: groups,
            fields: { value: "parentId" },
            cascadeTo: 'countries'
        });
        var sample2 = new ej.DropDownList($("#countries"), {
            dataSource: countries,
            fields: { value: "value" },

```



```

cascadeTo: 'capitalList',
enabled: false
});
var sample3 = new ej.DropDownList($("#capitalList"), {
dataSource: capital,
fields: { value: "value" },
enabled: false
});
});
}

```

First two DropDownList cascaded based on the parentId, and then from second to third, cascading performed based on the value field.

Select Continent

Asia

Select Country

Bangladesh

Select State

Dhaka

### Search

Items are searched based on the keyed in values to the textbox. There are two types of searches,

- Incremental Search
- Filter Search

#### Incremental Search

Selects the item in the popup list based on the keyed in value. If the time taken to type exceeds 1000 milliseconds then filtered items will be reset based on the current input value. By default this mode of search is enabled. Incremental search can be case sensitive or case insensitive. To make case sensitive, you can use [caseSensitiveSearch](#) property.

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```

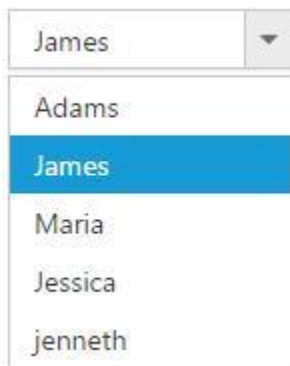
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
var items = [{
text: "Adams",
value: "emp1"
}, {
text: "James",
value: "emp2"
}, {
text: "Maria",
value: "emp3"
}, {

```

```

text: "Jessica",
value: "emp4"
}, {
text: "Jenneth",
value: "emp5"
}];
$(function() {
var sample = new ej.DropDownList($("#dropdown1"), {
dataSource: items,
fields: {
text: "text",
value: "value"
},
enableIncrementalSearch: true,
caseSensitiveSearch: true
});
});
}

```



#### Filter search

You can quickly locate specific item within a large data source by filtering matches with a search box. A text box appears in the popup list for searching when [enableFilterSearch](#) property is enabled. By default, filtering returns the matched items list based on text in search textbox.

You can configure the search filter by using [filterType](#) property. There is two types of filter options,

- Starts With
- Contains

**Note:** Items are filtered based on “contains” filter type by default.

#### HTML

```
<input type="text" id="dropdown1" />
```

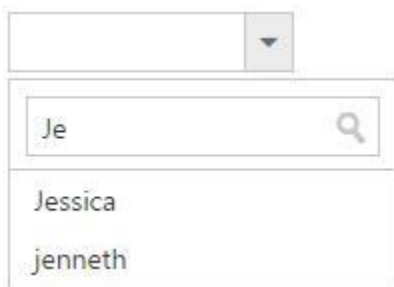
#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
var items = [{
text: "Adams",

```

```
value: "emp1"
}, {
text: "James",
value: "emp2"
}, {
text: "Maria",
value: "emp3"
}, {
text: "Jessica",
value: "emp4"
}, {
text: "Jenneth",
value: "emp5"
}];
$(function() {
var sample = new ej.DropDownList($("#dropdown1"), {
dataSource: items,
fields: {
text: "text",
value: "value"
},
enableFilterSearch: true,
filterType: "startsWith"
});
});
});
```



## Template Support

By default you can add any text or image to the DropDownList list item. To customize the items layout or to create your own visualized elements you can use this template support.

### Header Template

You can create the popup header by using [headerTemplate](#) property. You can add any HTML content in header template.

---

**Note:** Refer the check all option in popup list : [link](#)

---

### Template Field

Create a set of div containers with common syntax or elements and assign it to the [template](#) property. You can add any HTML mark-up element inside the DropDownList list using this property.

In the demo, a JSON array is created with text, imgId, role and country which is initialized with dataSource property. Content template is created by using the corresponding fields and assigned in

template property. The content template is customized with images and custom CSS styles to visualize the items in popup.

### HTML

```
<input type="text" id="dropdown1" />
```

### CSS

```
.imgId {  
margin: 0;  
padding: 3px 10px 3px 3px;  
border: 0 none;  
width: 60px;  
height: 60px;  
float: left;  
}  
.e-header {  
font-weight: bold;  
border-bottom: 1px solid #c8c8c8;  
background: #c8c8c8;  
}  
.e-header > span {  
display: inline-block;  
padding: 10px;  
}  
.ename {  
font-weight: bold;  
padding: 6px 3px 1px 3px;  
}  
.role, .cont {  
font-size: smaller;  
padding: 3px 3px -1px 0px;  
}
```

### JAVASCRIPT

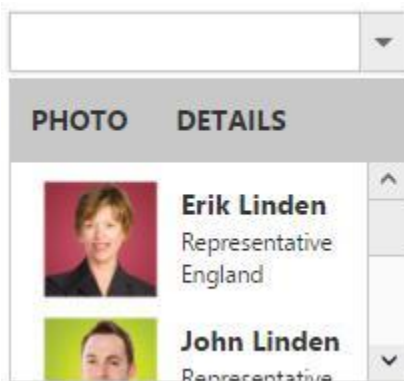
```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module DropDownListComponent {  
var List = [{  
text: "Erik Linden",  
imgId: "3",  
role: "Representative",  
country: "England"  
}, {  
text: "John Linden",  
imgId: "6",  
role: "Representative",  
country: "Norway"  
}, {  
text: "Louis",  
imgId: "7",  
role: "Representative",  
country: "Australia"  
}, {
```

```

text: "Lawrence",
imgId: "8",
role: "Representative",
country: "India"
}];
$(function() {
var sample = new ej.DropDownList($("#dropdown1"), {
width: 200,
dataSource: List,
headerTemplate: "<div class='e-header'><span>PHOTO</span>
<span>DETAILS</span></div>",
template: '<div>' + '<div class="ename"> ${text} </div><div class="role">
${role} </div><div class="cont"> ${country} </div></div>'
});
});
}

```

**Note:** Images for this sample are available in (installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\samples\web\themes\images<br/>



**Note:** Showing custom selected valued in the input field of DropDownList : [link](#)

## Customization

### Adding watermark text

It provides the short description of the expected value in dropdown and will display the text until any item is selected. You can set this text using [watermarkText](#) property.

### HTML

```
<input type="text" id="dropdown1" />
```

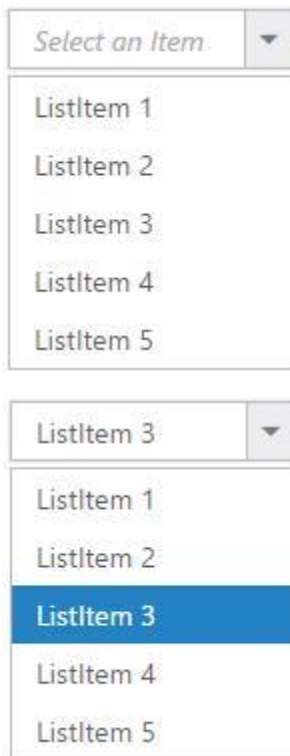
### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
var items = [{
text: "ListItem 1",
value: "item1"
}, {
text: "ListItem 2",

```

```
value: "item2"
}, {
text: "ListItem 3",
value: "item3"
}, {
text: "ListItem 4",
value: "item4"
}, {
text: "ListItem 5",
value: "item5"
}];
$(function () {
var sample = new ej.DropDownList($("#dropdown1"), {
dataSource: items,
fields: {
text: "text",
value: "value"
},
watermarkText: "Select an Item"
});
});
}
```



### Applying Rounded Corner

You can use [showRoundedCorner](#) property to add rounded borders to the input and popup elements. By default, rounded corner property is disabled in DropDownList.

### HTML

```
<input type="text" id="dropdown1" />
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  var items = [{
    text: "ListItem 1",
    value: "item1"
  }, {
    text: "ListItem 2",
    value: "item2"
  }, {
    text: "ListItem 3",
    value: "item3"
  }, {
    text: "ListItem 4",
    value: "item4"
  }, {
    text: "ListItem 5",
    value: "item5"
  }];
  $(function () {
    var sample = new ej.DropDownList($("#dropdown1"), {
      dataSource: items,
      fields: {
        text: "text",
        value: "value"
      },
      showRoundedCorner: true
    });
  });
}
```



**Information:** The browser support details for rounded corner is given [here](#).

**Enable/Disable the widget**

The [enabled](#) property is used to indicate whether the widget can respond to the user interaction or not. You can disable it by assigning false to this property. When the widget is disabled state, you cannot interact with the control.

**Note:** you can also use [enable\(\)](#) or [disable\(\)](#) public methods.

## HTML

```
<input type="text" id="dropdown1" />
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
    var items = [{
        text: "ListItem 1",
        value: "item1"
    }, {
        text: "ListItem 2",
        value: "item2"
    }, {
        text: "ListItem 3",
        value: "item3"
    }, {
        text: "ListItem 4",
        value: "item4"
    }, {
        text: "ListItem 5",
        value: "item5"
    }];
    $(function () {
        var sample = new ej.DropDownList($("#dropdown1"), {
            dataSource: items,
            fields: {
                text: "text",
                value: "value"
            },
            enabled: false
        });
    });
}
```



**Note:** you can disable/enable the single or multiple list items by using [disableItemsByIndices](#) and [enableItemsByIndices](#) property.

### Applying HTML Attributes

Additional HTML attributes can be applied to the widget by using [htmlAttributes](#) property. The valid attributes such as name, required, read-only and disabled are directly applied to the input element of DropDownList, and other attributes such as style, class will be applied to the outer wrapper element of DropDownList. Please refer to the [How-to](#) section.

**Note:** when you add an item dynamically to the widget, you can specify the HTML attributes in the [addItem\(\)](#) method parameters.

## HTML

```
<input type="text" id="dropdown1" />
```



## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  var items = [{
    text: "ListItem 1",
    value: "item1"
  }, {
    text: "ListItem 2",
    value: "item2"
  }, {
    text: "ListItem 3",
    value: "item3"
  }, {
    text: "ListItem 4",
    value: "item4"
  }, {
    text: "ListItem 5",
    value: "item5"
  }];
  $(function () {
    var sample = new ej.DropDownList($("#dropdown1"), {
      dataSource: items,
      fields: {
        text: "text",
        value: "value"
      },
      htmlAttributes: {
        style: "border:1px solid red;"
      }
    });
  });
};
```



## Setting dimensions

### Widget Sizing

#### *Fixed Size DropDownList widget*

You can customize the widget dimensions using [width](#) and [height](#) properties. Fixed size values can be specified in pixel or percentage values. By default the DropDownList wrapper will be assigned with "143px" width and "30px" height.

#### *Fixed size popup list*

You can customize the popup list dimensions using [popupWidth](#) and [popupHeight](#) properties. Fixed size values can be specified in pixel or percentage values. By default popup width is auto and popup height is "152px".

#### *Auto Sizing*

DropDownList is adaptive to mobile and web layout such that it is adjustable with screen resolution. The textbox will be rendered based on its parent containers dimensions on assigning 100% values to the

width property. Default value for popupWidth is auto, so when you assign 100% to popupWidth then it will be rendered based on specified range.

#### *Limit the number of items*

You can use [itemsCount](#) property to fetch only the specific number of items from the data source. To fetch the remaining items you can enable [virtual scrolling](#) support which loads the data on scrolling the data items in popup list.

---

**Note:** By default popup list is shown on DropDownList button click but you can display the list initially by enabling the [showPopupOnLoad](#) property. You can also use [showPopup\(\)](#) or [hidePopup\(\)](#) methods at run time to display or hide the popup list.

---

### **HTML**

```
<input type="text" id="dropdown1" />
```

### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  var items = [{
    text: "ListItem 1",
    value: "item1"
  }, {
    text: "ListItem 2",
    value: "item2"
  }, {
    text: "ListItem 3",
    value: "item3"
  }, {
    text: "ListItem 4",
    value: "item4"
  }, {
    text: "ListItem 5",
    value: "item5"
  }
  ];
  $(function () {
    var sample = new ej.DropDownList($("#dropdown1"), {
      dataSource: items,
      fields: {
        text: "text",
        value: "value"
      },
      itemsCount: 3,
      showPopupOnLoad: true
    });
  });
}
```



### Popup resizing

To show a resize handle in the popup list, use [enablePopupResize](#) property. You can customize the resize functionality by setting dimensions to the following properties.

|                                |  |
|--------------------------------|--|
| <a href="#">minPopupWidth</a>  | Default value is 0, once set you cannot resize below to the specified width      |
| <a href="#">maxPopupWidth</a>  | Default value is null, once set you cannot extend beyond to the specified width  |
| <a href="#">minPopupHeight</a> | Default value is 0, once set you cannot resize below to the specified height     |
| <a href="#">maxPopupHeight</a> | Default value is null, once set you cannot extend beyond to the specified height |

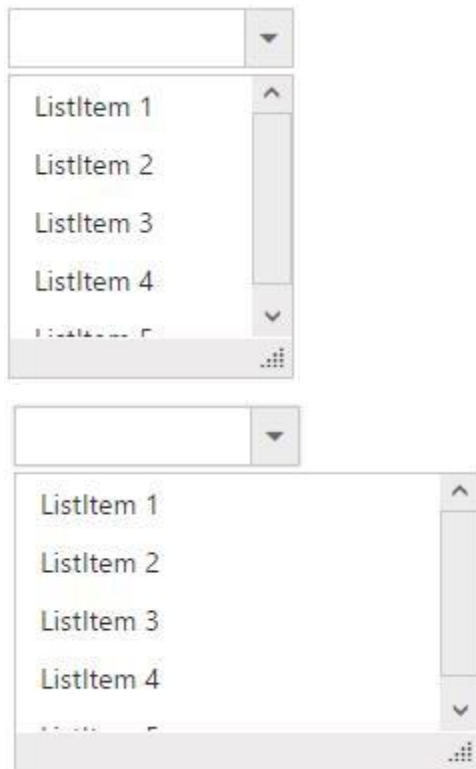
### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
    var items = [{
        text: "ListItem 1",
        value: "item1"
    }, {
        text: "ListItem 2",
        value: "item2"
    }, {
        text: "ListItem 3",
        value: "item3"
    }, {
        text: "ListItem 4",
        value: "item4"
    }, {
        text: "ListItem 5",
        value: "item5"
    }
    ];
    $(function () {
        var sample = new ej.DropDownList($("#dropdown1"), {
            dataSource: items,
            fields: {
```

```
text: "text",
value: "value"
},
enablePopupResize: true,
minPopupWidth: 150,
minPopupHeight: 150,
maxPopupWidth: 550,
maxPopupHeight: 550,
});
});
}
```



### State Persistence

DropDownList stores its model in local storage by defining the property [enablePersistence](#).

You can sustain the below given functionalities in DropDownList by enabling persistence property,

- selected items value in the textbox
- item's selection state in the popup list

Widget model will be stored in local storage / cookies of browser before page refreshes and reinitialized with the restored model after refresh.

The following properties are not included while maintaining DropDownList's state in local storage to keep localStorage compact.

- Fields
- Data source

- Query

### HTML

```
<form id="form1">
<input type="text" id="dropdown1" />
<input type="submit" value="Post" />
</form>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  var items = [{
    text: "Adams",
    value: "emp1"
  }, {
    text: "James",
    value: "emp2"
  }, {
    text: "Maria",
    value: "emp3"
  }, {
    text: "Jessica",
    value: "emp4"
  }, {
    text: "Jenneth",
    value: "emp5"
  }];
  $(function () {
    var sample = new ej.DropDownList($("#dropdown1"), {
      dataSource: items,
      fields: {
        text: "text",
        value: "value"
      },
      enablePersistence: true
    });
  });
}
```

### Accessing currently stored state

Persisted state can be accessed through local storage using corresponding key name. Key name formation would be in below order. It is combination of plugin name and control id.

### JAVASCRIPT

```
// DropDownList state as string
var dropdownlistStateString = window.localStorage.ejDropDownListdropdown;
//DropDownList state as object
var dropdownlistStateObject =
JSON.parse(window.localStorage.ejDropDownListdropdown);
```

**Note:** In the above example, 'ejDropDownList' is plugin name and 'dropdown' is control id.

### Localization

The DropDownList provides option to localize its strings, it is used to adapting the DropDownList to a particular local language. By default, the DropDownList will use the US English (en-US) as its language.

**Note:** The culture name has to be specified in a standard format such as [Language Code]-[County/Region Code].

To localize the dropdownlist's strings with your own localization, copy the default language informations and localize the strings in the values column. For example, to localize the DropDownList in German language ("de-DE").

Set the locale property of the DropDownList to the new language.

### JAVASCRIPT

```
<input type="text" id="selectCompany" />
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  $(function () {
    var dataList = ej.DataManager({ url:
      "http://mvc.syncfusion.com/services/Northwnd.svc/Customers" });
    var sample = new ej.DropDownList($("#selectCompany"), {
      dataSource: dataList,
      fields: { text: "CompanyName", value: 'ContactName' },
      width: 260,
      showCheckbox: true,
      locale: "de-DE",
      enableFilterSearch: true,
      enablePopupResize: true
    });
  });
  ej.DropDownList.Locales["de-DE"] = {
    emptyResultText: "Keine Vorschläge" //replace with your text
  };
}
```

### Load On Demand

Load on demand feature allows the DropDownList items load on request from the service/database, during only on click the DropDown icon or DropDownList. This functionality helps to improve performance on control initial rendering time. Because data items load on request.

The loadOnDemand property is used to enable or disable the load on demand functionality of the DropDownList.

### HTML

```
<input type="text" id="bikeList" />
```

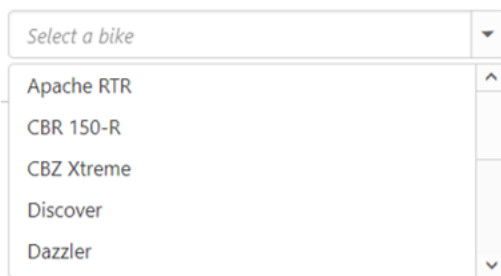
### JAVASCRIPT

```
module DropDownListComponent {
  var BikeList = [
```

```

{ id: "bk1", text: "Apache RTR" }, { id: "bk2", text: "CBR 150-R" }, { id:
"bk3", text: "CBZ Xterm" },
{ id: "bk4", text: "Discover" }, { id: "bk5", text: "Dazzler" }, { id:
"bk6", text: "Flame" },
{ id: "bk7", text: "Faze" }, { id: "bk8", text: "FZ-S" }, { id: "bk9", text:
"Pulsar" },
{ id: "bk10", text: "Shine" }, { id: "bk11", text: "R15" }, { id: "bk12",
text: "Unicorn" }
];
$(function () {
var sample = new ej.DropDownList($("#bikeList"), {
dataSource: BikeList,
width: "100%",
watermarkText: "Select a bike",
fields: { id: "id", text: "text", value: "text" },
loadOnDemand: true,
showRoundedCorner: true
});
});
}

```



## Accessibility

### Keyboard Navigation

You can use the keyboard short cut keys as an alternative to the mouse and interact with DropDownList widget.

Keyboard shortcut keys are,

| Key                            | Single selection  | Multi-selection   |
|--------------------------------|---|---|
| <a href="#">Access key</a> + j | Sets focus to the input   | Sets focus to the input   |
| Up                             | Selects the previous item when the popup is opened/closed   | Selects the previous item when the popup is opened/closed and clears all other selection  |
| Down                           | Selects the next item when the popup is opened/closed   | Selects the next item when the popup is opened/closed and clears all other selection      |
| Page up                        | Selects the item in next page (i.e.,) scrolls down to next set of items when the popup is opened/closed | Selects the item in next page (i.e.,) scrolls down to next set of items when the popup is |

|              |   |   |
|--------------|---|---|
|              |   | opened/closed and clear all existing selection  |
| Page down    | Selects the item in previous page (i.e.,) scrolls up to previous set of items when the popup is opened/closed | Selects the item in previous page (i.e.,) scrolls up to previous set of items when the popup is opened/closed and clear all existing selection  |
| Left arrow   | Selects the previous item when the popup is opened/closed. But in RTL mode selects next item                  | Selects the previous item when the popup is opened/closed and clears all other selection but in RTL mode selects the next item  |
| Right arrow  | Selects the next item when the popup is opened/closed. But in RTL mode selects previous item                  | Selects the next item when the popup is opened/closed and clears all other selection but in RTL mode selects the previous item  |
| Backspace    | No actions  | Deletes the selected items from last when visual mode is enabled. (i.e.,) on first backspace key press, the last item will be selected and second backspace press will remove the corresponding item. |
| Tab          | Close the popup if it is opened and focuses to next DOM element   | Close the popup if it is opened and focuses to next DOM element   |
| End          | Selects the last list item  | Select the last item and clear all selection  |
| Home         | Selects the first list item   | Select the first item and clear all selection   |
| Esc          | Close the popup if itâ€™s opened  | Close the popup if itâ€™s opened  |
| Space bar    | Close the popup is itâ€™s opened  | Check and uncheck the items which is focused.   |
| Delete       | No actions  | Deletes the focused item in textbox when visual mode is enabled   |
| Shift + up   | Selects the previous item when the popup is opened/closed   | Selects the immediate preceding item from current selected item   |
| Shift + down | Selects the next item when the popup is opened/closed   | Selects the immediate successive item from current selected item  |



|                     |   |  |
|---------------------|---|--|
| Shift + home        | Select the first list item  | Select all items between focused item to first item  |
| Shift + end         | Selects the last list item  | Select all items between focused item to last item   |
| Shift + page up     | Selects the item in previous page (i.e.,) scrolls down to previous set of items when the popup is opened/closed | Select all item between focused item to previous page or previous set of items when the popup is opened/closed |
| Shift + page down   | Selects the item in next page (i.e.,) scrolls up to next set of items when the popup is opened/closed           | Select all item between focused item to next page or next set of items when the popup is opened/closed         |
| Ctrl + up           | Selects the previous item when the popup is opened/closed   | Maintain current selections and hovers to the previous item when the popup is opened/closed                    |
| Ctrl + down         | Selects the next item when the popup is opened/closed   | Maintain current selection and hovers to the next item when the popup is opened/closed                         |
| Alt + down          | Opens the popup if itâ€™s closed  | Open the popup if itâ€™s closed  |
| Alt + up            | Closes the popup if itâ€™s opened   | Close the popup if itâ€™s opened   |
| Enter               | Closes the popup if itâ€™s opened   | Selects the focused item when ctrl key pressed otherwise closes the popup if itâ€™s opened                     |
| Shift + Tab         | Focuses the previous DOM element  | Focuses the previous DOM element   |
| Ctrl + shift + up   | Selects the previous item when the popup is opened/closed   | Maintains the existing items selection and selects the previous item from current selection.                   |
| Ctrl + shift + down | Selects the next item when the popup is opened/closed   | Maintain the existing items selection and selects next item from current selection.                            |

**Note:**

- i) For Ctrl and Shift operations with multi-selection, the current focused item should be selected.
- ii) No round robin method is used on navigating through the list items in pop up, so that navigation stops when reaching the start/end of the popup list.

## How To

Set focus to control initially?

[Access key](#) property can be added in input element to set focus. Here focus is set by using access key + "j".

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  var items = [
    { text: "ListItem 1", value: "item1" },
    { text: "ListItem 2", value: "item2" },
    { text: "ListItem 3", value: "item3" },
    { text: "ListItem 4", value: "item4" },
    { text: "ListItem 5", value: "item5" }
  ];
  $(function () {
    var sample = new ej.DropDownList($("#dropdown1"), {
      dataSource: items,
      fields: { text: "text", value: "value" }
    });
    //Control focus key
    $(document).on("keydown", function (e) {
      if (e.altKey && e.keyCode === 74) { // j- key code.
        $("#dropdown1_wrapper").focus();
      }
    });
  });
}
```

Clear the text of DropDownList input?

To clear the text of the DropDownList input, you can use [clearText](#) method.

Add an item dynamically to the DropDownList?

You can use [addItem](#) method to add single or multiple items dynamically to the popup list. You can define all the possible values that is supported by field property such as text, value, id, HTML attributes, selected, image and its associated attributes such as alt, width, and height etc.,

Adding text and value is demonstrated in the below given sample,

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
```

```
var items = [
{ text: "ListItem 1", value: "item1" },
{ text: "ListItem 2", value: "item2" },
{ text: "ListItem 3", value: "item3" },
{ text: "ListItem 4", value: "item4" },
{ text: "ListItem 5", value: "item5" }
];
$(function () {
var sample = new ej.DropDownList($("#dropdown1"), {
dataSource: items,
fields: { text: "text", value: "value" }
});
$('#dropdown1').ejDropDownList("addItem", { text: "New Text", value: "text1"
});
});
}
```

### Disable/ Enable the DropDownList widget?

You can enable or disable the DropDownList widget using "enabled" property or methods. Detailed information is given [here](#).

### Control the popup visibility via methods in script showPopup()/hidePopup()?

By default popup list is shown on DropDownList button click but you can display the list initially by enabling the [showPopupOnLoad](#) property. You can also use [showPopup\(\)](#) or [hidePopup\(\)](#) methods at run time to display or hide the popup list.

### Retrieve the selected item data from select event via arguments?

Bind the select event and you can retrieve the value from args.value.

### HTML

```
<input type="text" id="dropdown1" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
var items = [
{ text: "ListItem 1", value: "item1" },
{ text: "ListItem 2", value: "item2" },
{ text: "ListItem 3", value: "item3" },
{ text: "ListItem 4", value: "item4" },
{ text: "ListItem 5", value: "item5" }
];
$(function () {
var sample = new ej.DropDownList($("#dropdown1"), {
dataSource: items,
fields: { text: "text", value: "value" },
select: function (args) {
console.log("Value is " + args.value);
}
});
});
}
```

The following screenshot will exhibit the select event arguments details,



Append custom HTML in DropDownList popup outside the scroller part?

Create a custom HTML element and insert it after popup wrapper. Detailed sample is given [here](#)

Add check all option in popup list?

You can use [headerTemplate](#) property to add any HTML element. Code snippet to add check all option is given below,

### HTML

```
<input type="text" id="dropdown1" />
```

### CSS

```
.temp {
height: 30px;
display: block;
padding-left: 13px;
padding-top: 5px;
border-bottom: 1px solid #c8c8c8;
}
.e-chkbox-wrap .e-text {
font-size: 14px;
padding-left: 10px;
}
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
var items = [
{ text: "ListItem 1", value: "item1" },
{ text: "ListItem 2", value: "item2" },
{ text: "ListItem 3", value: "item3" },
{ text: "ListItem 4", value: "item4" },
{ text: "ListItem 5", value: "item5" }
];
$(function () {
var sample = new ej.DropDownList($("#dropdown1"), {
width: 300,
dataSource: items,
fields: { text: "text", value: "value" },
showCheckbox: true,
```

```

headerTemplate: "<div class='temp' ><input id ='check' type='checkbox' />
</div>"
});
$("#check").ejCheckBox({ text: "Check All", change: "Change" });
});
}
function Change(args) {
window.flag = true;
var obj = $("#dropdown1").ejDropDownList("instance");
if (args.isChecked) obj.checkAll();
else obj.uncheckAll();
window.flag = false;
}

```

The following screenshot exhibits the output of the above code,



To remove the items from DropDownList?

You can remove the items from the DropDownList by using [splice](#) method and then rebind the data source through set model.

Removing an entry from DropDownList is demonstrated in the below given sample.

### HTML

```
<input id="dropdown1" /> <button id="remove">Remove items</button>
```

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
var data = [
{ text: "ListItem1", value: "item1" },
{ text: "ListItem2", value: "item2" },
{ text: "ListItem3", value: "item3" },
{ text: "ListItem4", value: "item4" },
{ text: "ListItem5", value: "item5" }
];
// create DropDownList from input HTML element
$(function () {
var sample = new ej.DropDownList($("#dropdown1"), {
dataSource: data
});
});

```

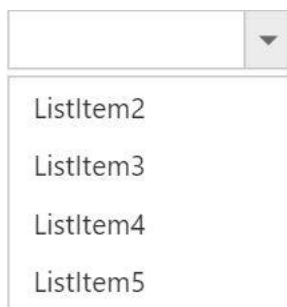
```
$("#remove").click(function () {  
    object = $("#dropdown1").data("ejDropDownList");  
    data1 = object.model.dataSource.splice(0);  
    data1.splice(0, 1);  
    object.setModel({ dataSource: data1 });  
});  
});  
}
```

The following screenshot exhibits the output of above code:

Before removing an item:



After removing an item:



Select the image rather than the text from the DropDownList when the template concept is used?

By default, the DropDownList displays only the text of the data item. We can able to customize the selected data to show case the custom visual element in the DropDownList's input element.

Initialize the DropDownList as follows

#### **JAVASCRIPT**

```
<input type="text" id="List " />  
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module DropDownListComponent {  
    //DataSource  
    var List = [  
        { text: "Erik Linden", image: "3", designation: "Representative", country:  
            "England" },  
        { text: "John Linden", image: "6", designation: "Representative", country:  
            "Norway" },  
    ]
```

```

{ text: "Louis", image: "7", designation: "Representative", country:
"Australia" },
{ text: "Lawrence", image: "8", designation: "Representative", country:
"India" }]];
//DropDownList Initialization
$(function () {
var sample = new ej.DropDownList($("#List"),{
dataSource: List,
fields: { text: "text", value: "image" },
width : "80px"
popupWidth: 200 ,
watermarkText: "Select an employee",
template: '<div>' +
'<div class="ename"> ${text} </div></div>',
select: "onSelect"
});
});
}

```

Upon selecting the items from the DropDownList, the client side event “select” will be triggered, in that find the input element which holds the text value and make it as “hidden” and then create the span element for the custom value and append to the DropDownList outer wrapper element.

#### JAVASCRIPT

```

function onSelect(args){
if(!args.model.showCheckbox && args.model.multiSelectMode == "none"){
var imgLocation = "http://js.syncfusion.com/demos/web/images/Employee/" +
args.value + ".png";
if($("#myImg").length != 1){
var target = $("#selectCar");
$(target).css({display : "none"});
var dateSpan = document.createElement('span');
dateSpan.innerHTML = '<img id="myImg" class="eimg" src=' + imgLocation + '
alt="employee"/>';
$(dateSpan).insertBefore(target);
}
else{
edit_save = document.getElementById("myImg");
edit_save.src = imgLocation;
}
}
}

```

Apply the following styles

#### JAVASCRIPT

```

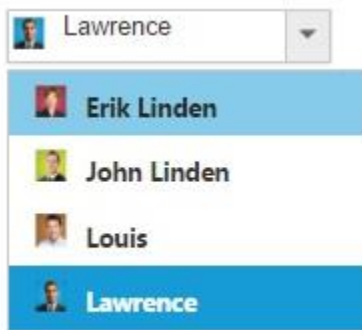
<style type="text/css" class="cssStyles">
.eimg {
margin: 0;
padding: 3px 10px 3px 3px;
border: 0 none;
width: 20px;
height: 20px;

```

```

float: left;
}
.ename {
font-weight: bold;
padding: 6px 3px 1px 3px;
}
.designation, .cont {
font-size: smaller;
padding: 3px 3px -1px 0px;
}
}
</style>

```



**Note:** This scenarios, will be suits for the single select mode in the DropDownList.

Apply HTML Attributes such as color and class directly to the input element rather than the outer wrapper element of DropDownList?

By default, htmlAttributes property of DropDownList, will add the HTML attributes to the input element of DropDownList. But, the following attributes such as class, style, readOnly and disabled cannot add directly to the input element.

This can be achieved, by adding the attributes directly to the input element if you needed.

### JAVASCRIPT

```

<input type="text" id="dropdown1" />
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
//Data Source
var List = [
{ text: "Erik Linden", role: "Representative", country: "England" },
{ text: "John Linden", role: "Representative", country: "Norway" },
{ text: "Louis", role: "Representative", country: "Australia" },
{ text: "Lawrence", role: "Representative", country: "India" }
];
//DropDownList Initialization
$(function () {
var data = new ej.DropDownList($("#List"), {
dataSource: List,
fields: { text: "text", value: "country" },
width: "200px"
});
});

```



```
data.element.attr("style", "background:yellow");
});
}
```



### Add tooltip on hovering the DropDownList's items?

In order to get tooltip on hovering the DropDownList popup items, use `htmlAttributes` field in it. Generate a `DataSource` with a field for mapping the `HtmlAttributes` having the "title" attribute value, which will allow you to show the tooltip on hovering. The `htmlAttributes` field will set the HTML properties for the list items.

#### JAVASCRIPT

```
<div class="control">
<div class="ctrl-label">Select a bike</div>
<input type="text" id="bikeList" />
</div>
<script type="text/javascript">
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
    BikeList = [
        { id: "bk1", text: "Apache RTR", tooltip: { title: "Apache RTR" } },
        { id: "bk2", text: "CBR 150-R", tooltip: { title: "CBR 150-R" } },
        { id: "bk3", text: "CBZ Xtreme", tooltip: { title: "CBZ Xtreme" } },
        { id: "bk4", text: "Discover", tooltip: { title: "Discover" } },
        { id: "bk5", text: "Dazzler", tooltip: { title: "Dazzler" } },
        { id: "bk6", text: "Flame", tooltip: { title: "Flame" } },
        { id: "bk7", text: "Fazzer", tooltip: { title: "Fazzer" } },
        { id: "bk8", text: "FZ-S", tooltip: { title: "FZ-S" } },
        { id: "bk9", text: "Pulsar", tooltip: { title: "Pulsar" } },
        { id: "bk10", text: "Shine", tooltip: { title: "Shine" } },
        { id: "bk11", text: "R15", tooltip: { title: "R15" } },
        { id: "bk12", text: "Unicorn", tooltip: { title: "Unicorn" } }
    ];
    $(function () {
        var data = new ej.DropDownList($("#bikeList"), {
            dataSource: BikeList,
            fields: { id: "id", text: "text", value: "text", htmlAttributes: "tooltip" }
        });
    });
}
<style class="cssStyles">
    .control {
        margin-left: 20px;
    }
    .ctrl-label {
        padding-bottom: 3px;
    }
</style>
```

### Stop/Prevent the events (change/select) in the DropDownList?

The client side events such as “select” or “change” can be prevented in the DropDownList by using argument name called cancel.

#### JAVASCRIPT

```
<div class="ctrl-label">Select a car</div>
<input type="text" id="selectCar" />
<div id="carsList">
<ul>
<li>Audi A4</li>
<li>Audi A5</li>
<li>Audi A6</li>
<li>Audi A7</li>
<li>Audi A8</li>
</ul>
</div>
<button id="button21">Select</button>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
$(function () {
var target = new ej.DropDownList($("#carsList"), {
targetID: "carsList",
width: "150px",
select: "onSelect",
change: "onSelect"
});
});
}
```

While selecting the items in the DropDownList, the select or change event will be triggered. In that, sets “true” to the cancel argument, which will prevent the further selecting of items in the DropDownList.

#### JAVASCRIPT

```
function onSelect(args) {
args.cancel = true;
}
```

### How can I add items in ejDropDownList at the first place in list?

To add the item in the certain position, then we should clear the old dataSource and add the items in array using jQuery Splice() method and then should re-initialize the dataSource in DropDownList.

Initialize the DropDownList as follows

#### JAVASCRIPT

```
<div class="dropdown">
<input type="text" id="dropdown" />
<div class="btn">
<button type="button" onclick="dataPrepend()">PREPEND</button>
<button type="button" onclick="dataAppend()">POSTPOND</button>
</div>
</div>
/// <reference path="tsfiles/jquery.d.ts" />
```

```

/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
  window.countries = [
    { text: "Algeria" },
    { text: "Argentina" },
    { text: "Armenia" },
    { text: "Brazil" },
    { text: "Bangladesh" }
  ];
  // Creates the DropDownList
  $(function () {
    var target = new ej.DropDownList($("#dropdown"), {
      dataSource: window.countries,
      field: { text: "text" }
    });
  });
}

```

Upon clicking to the Prepend button, which will insert the items at index of “0” in the DropDownList.

#### **JAVASCRIPT**

```

function dataPrepend() {
  var prepend = $('#dropdown').data("ejDropDownList");
  if (prepend.model.dataSource != null) {
    prepend.model.dataSource.splice(0, 0, { text: "India", value: "-1" });
    var b = prepend.model.dataSource;
    prepend.model.dataSource = null;
    prepend.option("dataSource", b);
  }
}

```

If you click the postpone button, which insert items at the last index in the DropDownList.

#### **JAVASCRIPT**

```

function dataAppend() {
  $('#dropdown').ejDropDownList("addItem", { text: "India" });
}

```

#### Can a DropDownList have delimiters in their JSON data source?

If the items have delimiter character, the same delimiter should not be set in the “delimiterChar” property of DropDownList. The default delimiter is “comma”. We suggest to use different delimiter character in the “delimiterChar” property of DropDownList if the multiSelectMode or showCheckbox is enabled.

Setting delimiter character other than comma will differentiate the selected items in DropDownList. Else you can use multiSelectMode as visualMode, so that each selected item in DropDownList will be boxed separately in the textbox.

Method 1: Setting custom delimiter Character

#### **JAVASCRIPT**

```

<input type="text" id="List" />

```

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
var List = [
{ text: "Erik, Linden", designation: "Representative" },
{ text: "John, Linden", designation: "Dancer"},
{ text: "Louis, John", designation: "Professor"},
{ text: "Lawrence, Joseph", designation: "Software Engineer" }
];
$(function () {
var target = new ej.DropDownList($("#List"),{
dataSource: List,
width: "100%",
fields: { text: "text", value : "designation" },
watermarkText: "Select an employee",
multiSelectMode: "delimiter",
delimiterChar : ";",
showCheckbox: true
});
});
}

```



## Method 2: Using Visual Mode

### JAVASCRIPT

```

<input type="text" id=" List " />
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module DropDownListComponent {
var List = [
{ text: "Erik, Linden", designation: "Representative" },
{ text: "John, Linden", designation: "Dancer" },
{ text: "Louis, John", designation: "Professor" },
{ text: "Lawrence, Joseph", designation: "Software Engineer" }
];
$(function () {
var target = new ej.DropDownList($("#List"),{
dataSource: List,
width: "100%",
fields: { text: "text", value: "designation" },
watermarkText: "Select an employee",
multiSelectMode: "visualmode",
showCheckbox: true
});
});
}

```

```
} 
```



## FileExplorer

### Overview

The **Essential JavaScript FileExplorer** is an interface for managing the File system through any web application. It allows the user to perform the most common file operations include browse, open, create, rename, copy, paste or move, delete and file searching.

For a live demo of FileExplorer check the online sample from [here](#)

---

**Note:** The FileExplorer control was officially added with the Essential Studio JavaScript package from the v13.1 release only.

---

### Key features

- Windows explorer like functionalities and appearances
- Handy file operations (copy, paste, move and delete)
- File download and upload
- File type restriction
- Easy UI customization
- Different layouts (grid and tile view)
- Context menu support
- built-in image viewer support
- Keyboard navigation
- Right to Left alignment (RTL) support
- Localization support

### Getting Started

Using the following steps, you can create a **Typescript** FileExplorer component.

#### Creating an FileExplorer in TypeScript

You can create a **Typescript** application with the help of the given [Typescript Getting Started Documentation](#).

Within an index.html file and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
```

```
<html>
<head>
<title>TypeScript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.web
.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add FileExplorer sample here-->
</body>
</html>
```

The FileExplorer can be created from a `div` element with the HTML `id` attribute and pre-defined options set to it.

### HTML

```
<div id="fileExplorer"></div>
<script src="app.js"></script>
```

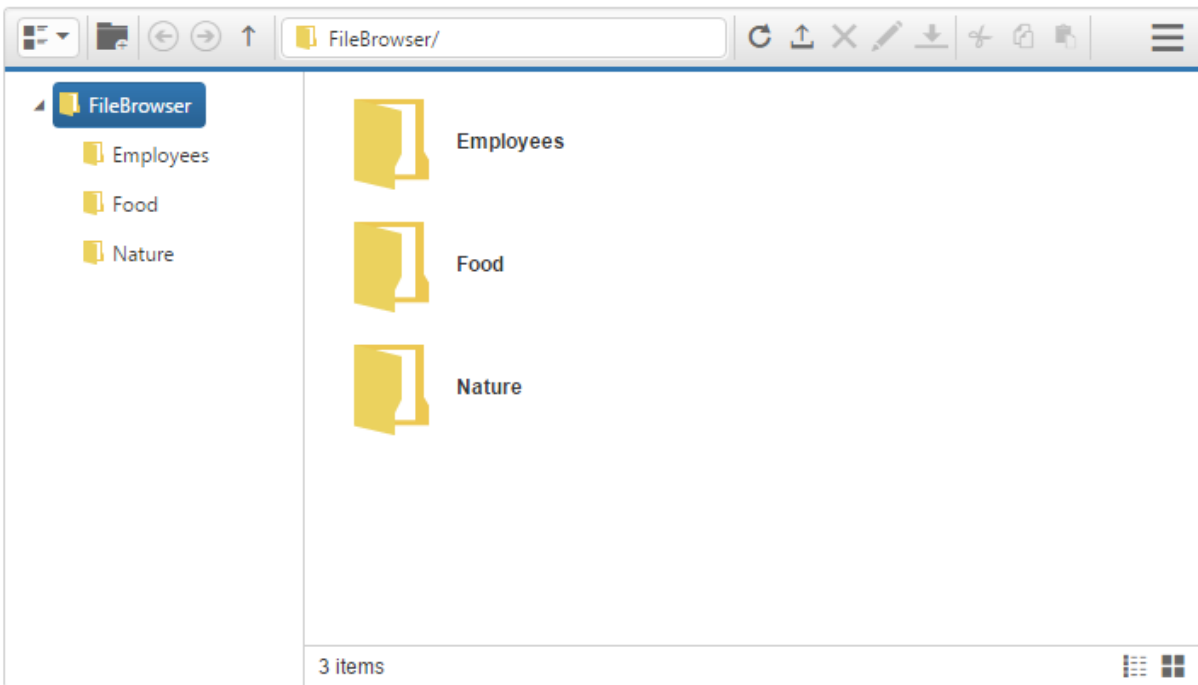
- Create `app.ts` file and use the below content

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var file = new ej.FileExplorer($("#fileExplorer"), {
path: (<any>window).baseurl + "Content/FileBrowser/",
width: "100%",
minWidth: "150px",
layout: "tile",
isResponsive: true,
ajaxAction: (<any>window).baseurl + "api/FileExplorer/FileOperations"
});
});
}
```

- Now build your application, so that the `app.ts` file will be compiled and automatically generate the `app.js` file which is added to your project (User has nothing to do with this file). Now, whatever code changes that you make in `app.ts` file will be reflected in `app.js` file by compiling and building the application.

Execution of the above code will render the following output.



## Resizing

- The FileExplorer has the resize support through the resize handle which appears at right-bottom corner of footer. By clicking the resize handle the user can resize the FileExplorer through the UI. While resizing the dimension of the FileExplorer is automatically adjusted.

The resize behavior can be enabled through the `enableResize` property.

## JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var file = new ej.FileExplorer($("#fileExplorer"), {
path: "http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/",
width: "100%",
minWidth: "150px",
layout: "tile",
isResponsive: true,
enableResize: true,
ajaxAction:
"http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations"
});
});
}
```

To perform the server side actions using local web API service, please refer the [link](#).

## Localization

The FileExplorer control provides the translations for all static texts and messages. By default the FileExplorer uses the US English (en-US) language, and it having the built-in values of all texts with the corresponding key.

The below table lists out the keys with corresponding texts in English culture.

### JAVASCRIPT

```
ej.FileExplorer.Locale["en-US"] = {
  Back: "Backward",
  Forward: "Forward",
  Upward: "Upward",
  Refresh: "Refresh",
  Addressbar: "Address bar",
  Upload: "Upload",
  Rename: "Rename",
  Delete: "Delete",
  Download: "Download",
  Error: "Error",
  Cut: "Cut",
  Copy: "Copy",
  Paste: "Paste",
  Details: "Details",
  Searchbar: "Search bar",
  Open: "Open",
  Search: "Search",
  EmptyResult: "No items match your search",
  EmptyFolder: "This folder is empty",
  NewFolder: "New Folder",
  Size: "Size",
  RenameAlert: "Please enter new name",
  NewFolderAlert: "Please enter new folder name",
  ContextMenuOpen: "Open",
  ContextMenuNewFolder: "New Folder",
  ContextMenuDelete: "Delete",
  ContextMenuRename: "Rename",
  ContextMenuUpload: "Upload",
  ContextMenuDownload: "Download",
  ContextMenuCut: "Cut",
  ContextMenuCopy: "Copy",
  ContextMenuPaste: "Paste",
  ContextMenuGetinfo: "Get info",
  ContextMenuRefresh: "Refresh",
  Item: "item",
  Items: "items",
  ErrorOnFolderCreation: "This destination already contains a folder named '{0}'. Do you want to merge this folder content with already existing folder '{0}'?",
  GeneralError: "Please see browser console window for more information",
  ErrorPath: "FileExplorer can't find '{0}'. Check the spelling and try again.",
  ReplaceAlert: "File named '{0}' already exists. Replace old file with new one?",
  DuplicateAlert: "There is already a file with the same name '{0}'. Do you want to create file with duplicate name",
```



```

DuplicateFileCreation: "There is already a file with the same name in this
location. Do you want to rename '{0}' to '{1}'?",
DeleteFolder: " Are you sure you want to delete ",
DeleteMultipleFolder: "Are you sure you want to delete these {0} items?",
CancelPasteAction: "The destination folder is a subfolder of source
folder.",
OkButton: "Ok",
CancelButton: "Cancel",
YesToAllButton: "Yes to all",
NoToAllButton: "No to all",
YesButton: "Yes",
NoButton: "No",
Grid: "Grid View",
Tile: "Tile View",
Name: "Name",
Location: "Location",
Type: "Item type",
Created: "Created",
Accessed: "Accessed",
Modified: "Modified",
DialogCloseToolTip: "close",
UploadSettings: {
  buttonText: {
    upload: "Upload",
    browse: "Browse",
    cancel: "Cancel",
    close: "Close"
  },
  dialogText: {
    title: "Upload Box",
    name: "Name",
    size: "Size",
    status: "Status"
  },
  dropAreaText: "Drop files or click to upload",
  filedetail: "The selected file size is too large. Please select a file
within the valid size.",
  denyError: "Files with #Extension extensions are not allowed.",
  allowError: "Only files with #Extension extensions are allowed.",
  cancelToolTip: "Cancel",
  removeToolTip: "Remove",
  retryToolTip: "Retry",
  completedToolTip: "Completed",
  failedToolTip: "Failed",
  closeToolTip: "Close"
}
};

```

### Change the Culture

The language of the FileExplorer can be changed by the [locale](#) property. This property allows the culture code of the country as input. The culture code has to specify in the standard format as **[Language Code]-[Country/Region Code]**.

For example the culture code for German is **de-DE**.

You can customize built-in text and messages based on your culture. The below example shows the usage of FileExplorer with German (de-DE) culture.

### JAVASCRIPT

```
ej.FileExplorer.Locale["de-DE"] = {
  Back: "rückwärts",
  Forward: "Nach Vorne",
  Upward: "nach oben",
  Refresh: "erfrischen",
  Addressbar: "Adressleiste",
  Upload: "hochladen",
  Rename: "umbenennen",
  Delete: "löschen",
  Download: "herunterladen",
  Error: "Fehler",
  Cut: "Schnitt",
  Copy: "Kopie",
  Paste: "kleben",
  Details: "Einzelheiten",
  Searchbar: "Searchbar",
  Open: "geöffnet",
  Search: "Suche",
  EmptyResult: "Keine Artikel entsprechen Ihrer Suche nach",
  EmptyFolder: "Dieser Ordner ist leer",
  NewFolder: "neuer Ordner",
  SelectedFileUrl: "Web-Adresse",
  SelectedFileName: "Titel",
  ImageWidth: "Breite",
  ImageHeight: "Höhe",
  Insert: "Insert",
  Cancel: "Rückgängig Machen",
  RenameAlert: "Bitte geben Sie neuen Namen",
  NewFolderAlert: "Geben Sie den neuen Ordnernamen ein",
  ContextMenuOpen: "geöffnet",
  ContextMenuNewFolder: "Neuer Ordner",
  ContextMenuDelete: "löschen",
  ContextMenuRename: "umbenennen",
  ContextMenuUpload: "hochladen",
  ContextMenuDownload: "Herunterladen",
  ContextMenuCut: "Schnitt",
  ContextMenuCopy: "Kopie",
  ContextMenuPaste: "kleben",
  ContextMenuGetinfo: "Ausführliche Infos",
  ContextMenuRefresh: "erfrischen",
  Item: " Artikel",
  Items: " Angebote",
  OkButton: "Ok",
  CancelButton: "Rückgängig machen",
  YesToAllButton: "Ja zu allem",
  NoToAllButton: "Nein, alle",
  YesButton: "Ja",
  NoButton: "Unterlassen Sie",
  Size: "Größe",
  Grid: "Gitter Ansicht",
  Tile: "Kachelansicht",
```

```

ErrorOnFolderCreation: "Dieses Ziel ist bereits ein Ordner mit dem Namen '{0}'. Möchten Sie diesen Ordner Inhalte mit bereits vorhandenen Ordner '{0}' zusammenführen möchten?",
GeneralError: "Bitte beachten Sie Browser Konsolenfenster für weitere Informationen",
ErrorPath: "FileExplorer kann nicht finden '{0}'. Überprüfen Sie die Schreibweise und versuchen Sie es erneut.",
ReplaceAlert: "Datei mit dem Namen '{0}' ist bereits vorhanden. Ersetzen Sie alte Datei durch eine neue?",
DuplicateAlert: "Es gibt bereits eine Datei mit dem gleichen Namen '{0}'. Möchten Sie diese Datei mit doppelten Namen erstellen möchten",
DuplicateFileCreation: "Es gibt bereits eine Datei mit dem gleichen Namen in diesem Ort. Möchten Sie umbenennen '{0}' bis '{1}' suchen?",
DeleteFolder: " Sind Sie sicher, dass Sie löschen möchten ",
DeleteMultipleFolder: "Sind Sie sicher, dass Sie diese {0} Einträge löschen?",
CancelPasteAction: "Der Zielordner ist ein Unterordner des Quellordners.",
Name: "Namen",
Location: "Ort",
Type: "Gegenstandsart",
Created: "Erstellt",
Modified: "geändert",
DialogCloseToolTip: "schließen",
UploadSettings: {
  buttonText: {
    upload: "hochladen",
    browse: "blättern",
    cancel: "Rückgängig Machen",
    close: "schließen"
  },
  dialogText: {
    title: "hochladen Box",
    name: "Name",
    size: "Größe",
    status: "Status"
  },
  cancelToolTip: "stornieren",
  removeToolTip: "entfernen",
  retryToolTip: "wiederholen",
  completedToolTip: "fertiggestellt",
  failedToolTip: "fehlgeschlagen",
  closeToolTip: "schließen"
}
};

```

## JAVASCRIPT

```

onChange(args) {
  $('#fileExplorer').ejFileExplorer('model.locale', args.value);
}

```

## Behavior Settings

Here are the some properties to customize the behavior of FileExplorer.

### File type restriction

FileExplorer control showcase all the files from the filesystem, here you can customize the file types that what you want to show by using [fileTypes](#) property. It filter the files based on the files extensions.

By default it having the value ".", so it allows all the file types. In case of image browser you can allow the image files only by setting ".png, .gif, .jpg, .jpeg", so it doesn't allow the other type of files.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var filePath =
"http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
var ajaxActionHandler =
"http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
var file = new ej.FileExplorer($("#fileExplorer"), {
path: filePath,
ajaxAction: ajaxActionHandler,
fileTypes: "*.png, *.gif, *.jpg, *.jpeg"
});
});
}
```

### Customize the AJAX request settings

As you already know FileExplorer is a client – server based control and each action performed in the client sends an AJAX request to the server to perform the server side operations. While the AJAX request, the AJAX configurations can be customized through [ajaxSettings](#) property.

The [beforeAjaxRequest](#) event will be triggered before the AJAX request is performed. You can modify the ajax request in this event.

You can see the following requests passed during the **client – server actions**:

- Read,
- Create folder
- Remove
- Rename
- Paste
- Get details
- Upload
- Download
- Get Image

The following is the syntax for ajaxSettings property.

### JAVASCRIPT

```
ajaxSettings: { read: {}, createFolder: {}, remove: {}, rename: {}, paste:
{}, getDetails: {}, upload: {}, download: {}, getImage: {} }
```

The actions “read, createFolder, remove, rename, paste, getDetails” are supported all the jQuery AJAX configurations. The remaining actions “upload, download, getImage” are accepted the URL only.

If you want to customize read action alone, the AJAX **url** and **dataType** are changed for the read action, refer the below code example.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var filePath =
"http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
var ajaxActionHandler =
"http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
var ajaxJSONPActionHandler =
"http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperationsCo
rs";
var file = new ej.FileExplorer($("#fileExplorer"), {
path: filePath,
ajaxAction: ajaxActionHandler,
ajaxSettings: {
read: {
url: ajaxJSONPActionHandler,
dataType: "jsonp"
}
}
});
});
}
```

For following file actions, you have to make the custom request in URL format only, not able to use jQuery AJAX configurations as like "dataType", "contentType", "async", etc..

- upload
- download
- getImage

Also its compulsory to add "{0}" in the end of "upload, download and getImage" URL's. That helps to pass the internal parameters with the action request.

#### JAVASCRIPT

```
ajaxSettings: {
upload: {
url: ajaxActionHandler + "/upload{0}"
},
download: {
url: ajaxActionHandler + "/download{0}"
},
getImage: {
url: ajaxActionHandler + "/getImage{0}"
}
}
```

---

**Note:** `ajaxActionHandler` - prefix URL of AJAX handling method

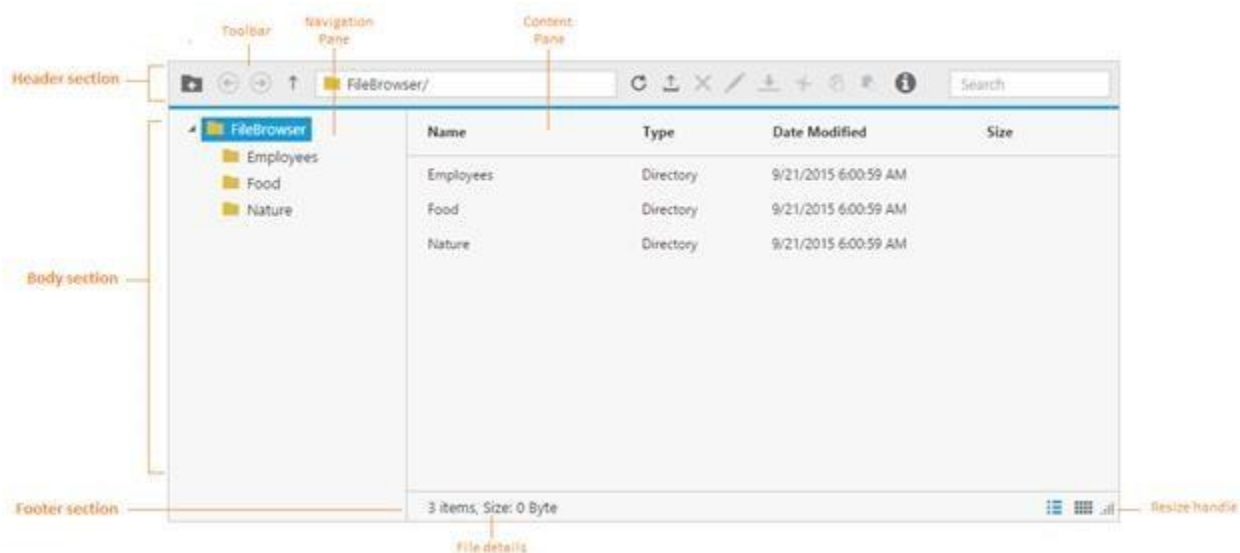
---

## User Interface

FileExplorer UI combined of several sections like header, body and footer. Each section having the corresponding components to perform the FileExplorer related operations.

- The **header section** contains the toolbar which having the list of tools to perform the file and navigation related operations.
- The **body section** is the main area which explores the filesystem contents. It separated into
- **Navigation pane** which contains the tree view to showcase the folder hierarchy
- **Content pane** which show cases the files from the file system. The files in the content pane can be viewable in the following modes
- Grid
- Tile
- The **footer section** contains the status bar which showcases the file details.

The following screenshot shows the diagrammatic detail of FileExplorer UI:



From above screenshot, you can see FileExplorer has several subcomponents for different functionalities. The upcoming sections explain the brief details of each component and their customizations.

## File Actions

The FileExplorer provides the support for file and folder related operations and these can be done through the toolbar items as well as through the context menu items.

### File action through Toolbar

The toolbar element is placed in the Header section, and it having the list of tools to perform the file, navigation and search related operations.

For more details about file action through toolbar, please refer [here](#).

### File action through Context menu

The FileExplorer having the support for context menu which opens while right click on any file or folder. The context menu having the set of items which is used to perform the corresponding operations in a handy pic way. For example the user can delete the file through the context menu by right click on it.

In FileExplorer, in the following places the context menu appears.

- While right click on treeview nodes (from navigation pane)
- While right click on File / Folder
- While right click on layout (content pane)

For more details about file action through context menu, please refer [here](#).

### Toolbar

The toolbar element having the number of tools, and each tool can be configurable.

#### Toolbar items

The toolbar having the list of items to perform various operations and grouped into some categorizes.

From below you can see the available toolbar items and its categorization:

#### JAVASCRIPT

```
// all tools grouped under the below categories
tools: {
  creation: ["NewFolder"],
  navigation: ["Back", "Forward", "Upward"],
  addressBar: ["Addressbar"],
  editing: ["Refresh", "Upload", "Delete", "Rename", "Download"],
  copyPaste: ["Cut", "Copy", "Paste"],
  getProperties: ["Details"],
  searchBar: ["Searchbar"],
  layout: ["Layout"],
  sortBy: ["SortBy"]
},
// all tools categories are placed in the toolbar by the below order
toolsList: ["layout", "creation", "navigation", "addressBar", "editing",
"copyPaste", "sortBy", "getProperties", "searchBar"]
```

The following table explains the functionality of each toolbar item:

| Tool name | Description  |
|-----------|--|
| NewFolder | It creates a new folder on the current directory.<br><br>While click on the NewFolder item, the dialog displays to get the folder name. Based on the user input, FileExplorer creates new folder on the current directory. Also <a href="#">createFolder</a> event will be triggered when new folder is created successfully in the file system. |
| Back      | It navigates to the previous directory from the user history. When the backward history is not available when it is disabled state.  |

|             |   |
|-------------|---|
| Forward     | It navigates to the next directory from the user history. When the forward history is not available when it is disable state.   |
| Upward      | It navigates to the parent directory of the current folder.   |
| Address bar | <p>The Address bar is the textbox which displays the path of the current directory, as a series of its parent directory separated by slash (â€œ/â€•).</p> <p>And the address bar is an editable area, so you can enter any directoryâ€™s path to a quick navigation.</p>                        |
| Refresh     | It refreshes the current directory.   |
| Upload      | <p>It uploads a file or list of files into the current directory.</p> <p>And you can customize the upload configurations, for details check <a href="#">here</a>.</p>   |
| Delete      | <p>It delete the current selected file or folder. The delete icon is enable state if you select any file or folder.</p> <p>If you selected the multiple files, it deletes all the selected items.</p>   |
| Rename      | <p>This is used to rename the current selected file or folder. The rename icon is enable state if you select any file or folder.</p> <p>Even if you selected multiple files, it renames the last selected file only.</p>  |
| Download    | <p>It downloads the selected files. The download icon is enable state if you select any file or folder.</p> <p>The <a href="#">beforeDownload</a> event will be triggered before the files are downloaded.</p> <p>If you select multiple files, it downloads all the files in a zip format.</p> |
| Cut         | It makes the copy of the selected files or folders into the clipboard. When the user paste the files in any location, the files are removed from the source location.The <a href="#">cut</a> event will be triggered when files or folders are removed from the source.                         |
| Copy        | It makes the copy of the selected files or folders into the clipboard. When the user paste the files, the copy of the files are pasted in the target location.The <a href="#">copy</a> event will be triggered when file or folder is copied.   |
| Paste       | <p>It pastes the files from the clipboard into the currently selected folder.</p> <p>Note: Only when the files are copied into the clipboard it is enabled.The <a href="#">paste</a> event will be triggered when file or folder is pasted.</p>   |



|            |   |
|------------|---|
|            |   |
| Details    | It displays the details of the current selected file or folder.   |
| Search bar | <p>The Search bar is the textbox which is used to search the files from the current directory. It list the files based on the user search.</p> <p>The search behavior of the Search bar can be customize, for details check <a href="#">here</a>.</p> |
| Sort By    | It's used to sorting the files and folders from the current directory. The sorting can be done based on the columns available from grid, in both ascending and descending order.  |

### Toolbar Visibility

The visibility of the toolbar can be customized through the [showToolbar](#) property. By disabling this property you can remove the toolbar from FileExplorer. Also you can remove the particular toolbar item by using [removeToolbarItem](#) method.

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var filePath =
"http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
var ajaxActionHandler =
"http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
var file = new ej.FileExplorer($("#fileExplorer"), {
path: filePath,
ajaxAction: ajaxActionHandler,
// hides the toolbar
showToolbar: false
});
});
}

```

### Toolbar Configuration

As you can see the available toolbar items from [here](#). From the list of available items, you can configure and group your required toolbar items only through the [tools](#) property. And also you can arrange the toolbar items by the [toolsList](#) property.

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var filePath =
"http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
var ajaxActionHandler =
"http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
var file = new ej.FileExplorer($("#fileExplorer"), {

```

```

path: filePath,
ajaxAction: ajaxActionHandler,
// denotes the order of the tools categories
toolsList: ["creation", "navigation", "addressBar", "copyPaste",
"searchBar"],
// mentions the required tool items under each category
tools: {
creation: ["NewFolder"],
navigation: ["Back", "Forward", "Upward"],
addressBar: ["Addressbar"],
copyPaste: ["Cut", "Copy", "Paste"],
searchBar: ["Searchbar"]
}
});
});
}

```

### Search bar

The Search bar can be customized through the [filterSettings](#) property. By default the search doesn't consider the case sensitivity, and the search works based on [filter type](#).

The FileExplorer allows the following filter types in the search functionality.

- Starts with
- Contains
- Ends with

### JAVASCRIPT

```

ej.FileExplorer.filterType = {
/** Supports to search text with startswith */
StartsWith: "startswith",
/** Supports to search text with contains */
Contains: "contains",
/** Supports to search text with endswith */
EndsWith: "endswith",
};

```

So you can configure the filter type with case sensitivity like below:

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var filePath =
"http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
var ajaxActionHandler =
"http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
$("#fileExplorer").ejFileExplorer({
path: filePath,
ajaxAction: ajaxActionHandler,
filterSettings: {

```

```
// it enables the case sensitive search
caseSensitiveSearch: true,
// it changes the search filter type as "startswith"
filterType: ej.FileExplorer.filterType.StartsWith
}
});
});
}
```

### Custom Tool in Toolbar

From the [toolbar items](#) you can see the list of built-in tools to perform the operations. Along with this built-in tools, you can add your custom tool with the custom functionality.

You can find an online demo sample of FileExplorer with custom tool from [here](#).

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var filePath =
"http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
var ajaxActionHandler =
"http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
var file = new ej.FileExplorer($("#fileExplorer"), {
path: filePath,
ajaxAction: ajaxActionHandler,
// denotes the order of the tools categories
toolsList: ["creation", "navigation", "addressBar", "copyPaste",
"searchBar", "customTool"],
// defines the tool items for customTool category
tools: {
customTool: [{
name: "Help",
tooltip: "Help ",
css: "e-fileExplorer-toolbar-icon Help",
action: function () {
// handle the click handler of custom tool
alert("custom tool item clicked");
}
}]
}
});
});
}
```

Define the CSS that needs to apply for the custom tool.

### CSS

```
.e-fileExplorer-toolbar-icon {
height: 22px;
width: 22px;
font-family: 'ej-webfont';
font-size: 18px;
```

```
margin-top: 2px;
text-align: center;
}
.e-fileExplorer-toolbar-icon.Help:before {
content: "\e72b";
}
```

### Enable / Disable the Toolbar Item

Each toolbar item can be enabled or disabled through the below client-side public methods.

- [enableToolbarItem](#)
- [disableToolbarItem](#)

These methods accept the tool name as the parameter. It also allows the parameter as tool item index or the jQuery object of the tool item.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var fileExpObj = new ej.FileExplorer($("#fileExplorer"), {
});
// this disables the NewFolder item
fileExpObj.disableToolbarItem("NewFolder");
// this also disables the NewFolder item (since index of NewFolder is 0)
fileExpObj.disableToolbarItem(0);
// this enables the NewFolder item
fileExpObj.enableToolbarItem("NewFolder");
});
}
```

### Enable / Disable the custom added tool in Toolbar Item

If you want to enable / disable the custom added tool in toolbar, you need to pass the corresponding li elements of custom added tool in [enableToolbarItem](#) / [disableToolbarItem](#) method of FileExplorer. Since we have considered this custom tool as an object type.

### JAVASCRIPT

```
var fileExpObj = $("#fileExplorer").data("ejFileExplorer");
//tool is a cssClass of FileExplorer
// this disables the custom tool item
var li = $(".tool").find(".Help").closest('li');
fileExpObj.disableToolbarItem(li);
// this enables the custom tool item
fileExpObj.enableToolbarItem(li);
```

### Customizing the Upload Functionality

FileExplorer helps you to upload the file using [Upload](#) component. File upload can be done through the toolbar item or context menu item. The [uploadSettings](#) property is used to configure the upload functionalities.

This property has the below sub properties with the default values:

### JAVASCRIPT

```
uploadSettings: {
  allowMultipleFile: true,
  maxFileSize: 31457280,
  autoUpload: false
}
```

**allowMultipleFile:** This property used to control the behavior of multiple files upload and this was enabled by default so you can upload multiple files at a time. If you disable this allowMultipleFile property then you can upload only one file at a time.

**maxFileSize:** The property limits the maximum file size to upload. It accepts the value in bytes.

**autoUpload:** when you enable this property, the upload action performed automatically after select the files. When you disable this property, it shows a confirmation dialog with the selected file details and perform the upload action on press the "upload" button.

During upload process following events will be triggered, [beforeUploadSend](#), [beforeUploadDialogOpen](#), [beforeUpload](#), [uploadError](#), [uploadSuccess](#) and [uploadComplete](#). You can customize the upload settings with these events

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
  $(function () {
    var filePath =
      "http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
    var ajaxActionHandler =
      "http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
    var file = new ej.FileExplorer($("#fileExplorer"), {
      path: filePath,
      ajaxAction: ajaxActionHandler,
      uploadSettings: {
        // it disables the multi file upload functionality
        allowMultipleFile: false,
        // it enables the auto upload functionality
        autoUpload: true
      }
    });
  });
}
```

## Context Menu

The context-menu has [list of items](#) which helps to perform FileExplorer operations, and it appears based on the target such as file or folder.

### Context menu items

The below table shows the context menu items corresponding to the location where it is opened:

| Context menu location | Context menu items | Screenshot |
|-----------------------|--------------------|------------|
|-----------------------|--------------------|------------|

|  |   |  |
|--|---|--|
| While right click on treeview nodes (from navigation pane) | <i>New folder</i><br>Upload<br>Delete<br>Rename<br>Cut<br>Copy<br>* Paste                                 |  |
| While right click on File / Folder                         | <i>Open</i><br>Download<br><i>Upload</i><br>Delete<br><i>Rename</i><br>Cut<br>Copy<br>Paste<br>* Get info |  |
| While right click on layout (content pane)                 | <i>Refresh</i><br>Paste<br><i>Sort By</i><br>New folder<br><i>Upload</i><br>Get info                      |  |

The below table explains the behavior of each context menu item:

|          |   |
|----------|---|
| Open     | It opens the selected folder. When an image file selected it opens the preview of the image. For the remaining files this option becomes disabled.  |
| Download | It downloads the selected file. When a file or number of files selected at that time only download option enabled.<br><br>If multiple files selected then it downloads all the files in a zip format. |
| Cut      | It makes the copy of the selected files or folders into the clipboard. When the user paste the files in any location, the files are removed from the source location.                                 |
| Copy     | It makes the copy of the selected files or folders into the clipboard. When the user paste the files, the copy of the files only pasted in the target location.                                       |
| Paste    | It paste the files from the clipboard into the current selected folder. Note that when the files are copied into the clipboard at that time only it enabled.  |
| Delete   | It deletes the current selected file or folder. When you select any file or folder at that time only this option gets enabled.  |

|            |   |
|------------|---|
|            | If multiple files selected then it deletes all the selected items.  |
| Rename     | This is used to rename the current selected file or folder. When you select any file or folder at that time only this option gets enabled.<br><br>Even multiple files selected it renames the single file only. |
| New folder | It creates a new folder on the current directory.<br><br>While click on the NewFolder item a dialog appears to get the folder name. Based on the user input, a new folder create on the current directory.      |
| Upload     | It uploads a file or list of files into the current directory.  |
| Get info   | It displays the details of the current selected file or folder.   |
| Sort By    | It's used to sorting the files and folders from the current directory. The sorting can be done based on the columns available from grid, in both ascending and descending order.                                |

### Context menu Visibility

The presence of the context menu can be controlled by the [showContextMenu](#) property. This was enabled by default, and by disabling this property you can prevent our built-in context menu.

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
  $(function () {
    var filePath =
      "http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
    var ajaxActionHandler =
      "http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
    var file = new ej.FileExplorer($("#fileExplorer"), {
      path: filePath,
      ajaxAction: ajaxActionHandler,
      // hides our built-in context menu
      showContextMenu: false
    });
  });
}

```

### Enable / Disable the Context menu Item

The context menu items can be enabled or disabled through the client side public methods. It enables or disables the item from all the context menu where it is present.

For example, if you disable the “Upload” item, it disables in all places wherever it appears such as open the context menu on treeview, open on file/folder, and open on layout.

- [enableMenuItem](#)
- [disableMenuItem](#)

These methods only accepts the context menu item name as the parameter.

### JAVASCRIPT

```
$(function () {  
    var fileExpObj = new ej.FileExplorer($("#fileExplorer"), {  
    });  
    // this disables the New Folder item  
    fileExpObj.disableMenuItem("New folder");  
    // this disables the Download item  
    fileExpObj.disableMenuItem("Download");  
});
```

### Context Menu Events

You would be notified with events when you try to open the context menu items (**menuBeforeOpen**), after context menu items is opened (**menuOpen**) and when you click the menu items (**menuClick**). The following code example illustrates how to define those events.

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module ExplorerComponent {  
    $(function () {  
        var file = new ej.FileExplorer($("#fileExplorer"), {  
            menuBeforeOpen: function (args) {  
                //you add/remove the context menu items in run time  
                //do your custom action here.  
                args.dataSource.pop();  
            },  
            menuOpen: function (args) {  
                //you can also identify which context menu is opened by  
                if (args.contextMenu == "cwd") {  
                    //do your custom action here.  
                }  
            },  
            menuClick: function (args) {  
                switch (args.text) {  
                    case "largeicons":  
                        //do your custom action here.  
                        break;  
                }  
            }  
        });  
    });  
}
```



## Customization

### Dimension Customization

The dimension of the FileExplorer can be customized through [height](#) and [width](#) property. The dimension can be set in percentage (ex; width: "100 %"), so that the control inherits the width from the parent element.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
  $(function () {
    var filePath =
      "http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
    var ajaxActionHandler =
      "http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
    var file = new ej.FileExplorer($("#fileExplorer"), {
      path: filePath,
      ajaxAction: ajaxActionHandler,
      height: "300px",
      width: "900px"
    });
  });
}
```

### Customizing the Navigation pane

The navigation pane contains the tree view element which displays all the folders from the filesystem in a hierarchical manner. This is useful to a quick navigation of any folder in the filesystem.

The visibility of the navigation pane can be controlled by the [showNavigationPane](#) property. By disabling this property, you can hide the navigation pane from FileExplorer.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
  $(function () {
    var filePath =
      "http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
    var ajaxActionHandler =
      "http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
    var file = new ej.FileExplorer($("#fileExplorer"), {
      path: filePath,
      ajaxAction: ajaxActionHandler,
      // hides the left side navigation pane
      showNavigationPane: false
    });
  });
}
```

### Customizing the Content pane

The content pane is the main part of the FileExplorer UI which displays all the files and folders from the filesystem. The content pane supports the following two types of layout views:

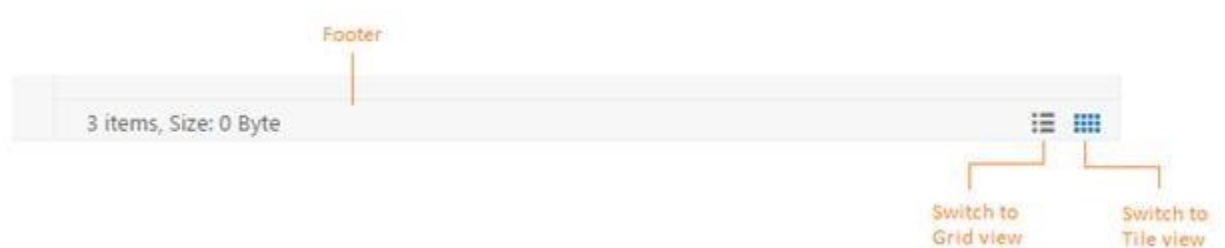
- Grid
- Tile

The **grid view** displays the files and folders in a grid layout with the details in separate columns. By default the grid view having the four columns which displays the file name, type, date modified and size of the file. For more details about grid view customization, refer [here](#).

The **tile view** display the files and folders like a small size icons. It allows the thumbnails for the image files so that you can view the tiny preview of all image files.

#### Changing the Layout views

You can change the layout of current view by the switcher which displays at right-bottom of footer in the FileExplorer. By clicking the grid and tile view buttons you can change the layout of current view.



Also the layout views can be changed through the [layout](#) property. The [layoutChange](#) event will be triggered whenever the layout view type is changed.

#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var filePath =
"http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
var ajaxActionHandler =
"http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
var file = new ej.FileExplorer($("#fileExplorer"), {
path: filePath,
ajaxAction: ajaxActionHandler,
// while rendering it displays as tile view
layout: ej.FileExplorer.layoutType.Tile
});
});
}

```

#### Customizing the Grid view

By default sorting is enabled in grid view of FileExplorer, it helps you to sort each columns in ascending or descending order by pressing the corresponding column header. The sorting functionality can be disabled by setting [allowSorting](#) property to false.

The behavior of the columns can be customized through the [columns](#) property.

#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />

```

```

/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var filePath =
"http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/"
var ajaxActionHandler =
"http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
var file = new ej.FileExplorer($("#fileExplorer"), {
path: filePath,
ajaxAction: ajaxActionHandler,
// under gridSettings you can customize the grid view functionalities
gridSettings: {
// disables the sorting functionality in grid view
allowSorting: false,
// it displays 3 columns only, like this you can display your // desired
columns here
columns: [
{ field: "name", headerText: "Name", width: 150 },
{ field: "dateModified", headerText: "Date Modified", width: 150 },
{ field: "size", headerText: "Size", width: 90, textAlign: "right",
headerTextAlign: "left" }
]
}
});
});
}

```

### Footer Customization

The footer displays the details of the current selected files and folders, and the footer contains the switcher to change the layout view. The visibility of the footer can be customized by the [showFooter](#) property.

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var filePath =
"http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
var ajaxActionHandler =
"http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
var file = new ej.FileExplorer($("#fileExplorer"), {
path: filePath,
ajaxAction: ajaxActionHandler,
// it hides the footer element
showFooter: false
});
});
}

```

### Customize the Root Folder name in FileExplorer

You can set the alias name to the root folder of FileExplorer by using `rootFolderName` API. It is used to replace the actual root folder name in the FileExplorer UI. Refer to the below code to set the alias name for the root folder of FileExplorer.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var filePath =
"http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
var ajaxActionHandler =
"http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
var file = new ej.FileExplorer($("#fileExplorer"), {
path: filePath,
ajaxAction: ajaxActionHandler,
rootFolderName: "This PC"
});
});
}
```

### Multiple Selection

The FileExplorer allows the user to select multiple files by enabling the [allowMultiSelection](#) property. The multiple selection can be done by pressing the **Ctrl** key or **Shift** key. You can select all the files in the directory by pressing "**Ctrl + A**". The multiple selection is useful on copy pasting multiple files, deleting multiple files and downloading multiple files. In another way, multiple files can be selected by mouse down and drag over the desired files. In addition to that, additional files can be selected with the preselected file by holding the ctrl/shift key and drag the mouse over the files. Also, holding the ctrl/shift key and dragging over selected files will unselect the selected files.

The [select](#) event will be triggered when the items of FileExplorer control is selected.

Also [unselect](#) event will be triggered when the items of FileExplorer control is unselected.

---

**Note:** For selecting files by mouse down and drag set `allowMultiSelection` property as true.

---

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var filePath =
"http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
var ajaxActionHandler =
"http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
var file = new ej.FileExplorer($("#fileExplorer"), {
path: filePath,
ajaxAction: ajaxActionHandler,
allowMultiSelection: true //allowMultiSelection property is true by default
});
});
}
```

## Resizing

The FileExplorer has the resize support through the resize handle which appears at right-bottom corner of footer. By clicking the resize handle the user can resize the FileExplorer through the UI. While resizing the dimension of the FileExplorer is automatically adjusted.

The resize behavior can be enabled through the [enableResize](#) property.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
  $(function () {
    var filePath =
      "http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
    var ajaxActionHandler =
      "http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
    var file = new ej.FileExplorer($("#fileExplorer"), {
      path: filePath,
      ajaxAction: ajaxActionHandler,
      // it enables the resize functionality
      // and a resize handle added in the Footer element
      enableResize: true
    });
  });
}
```

## Responsiveness

By enabling the [isResponsive](#) property, you can make the FileExplorer as responsive. While resizing the FileExplorer component, the inner content and toolbar items are automatically adjusted to equalize the size. The toolbar items are displayed within Dropdown on enabling the responsive. Otherwise it floated to the next line to equalize the space.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
  $(function () {
    var filePath =
      "http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
    var ajaxActionHandler =
      "http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
    var file = new ej.FileExplorer($("#fileExplorer"), {
      path: filePath,
      ajaxAction: ajaxActionHandler,
      // it enables the resize functionality
      enableResize: true,
      // it enables the control responsiveness
      isResponsive: true
    });
  });
}
```

### Restriction on Resize

You can restrict the dimension of the FileExplorer by setting the [minHeight](#), [maxHeight](#) and [minWidth](#), [maxWidth](#) properties while resizing the FileExplorer.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
  $(function () {
    var filePath =
      "http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
    var ajaxActionHandler =
      "http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
    var file = new ej.FileExplorer($("#fileExplorer"), {
      path: filePath,
      ajaxAction: ajaxActionHandler,
      enableResize: true,
      isResponsive: true,
      // restricts the dimensions for height and width
      minHeight: "200px",
      maxHeight: "400px",
      minWidth: "300px",
      maxWidth: "1200px"
    });
  });
}
```

### Drag and Drop Support

The FileExplorer allows files to be moved from one folder to another by using drag and drop. It also supports uploading a file by dragging it from Windows Explorer to a folder in the FileExplorer control.

You can enable or disable this support by using “**allowDragAndDrop**” API of FileExplorer.

The [dragStart](#), [drag](#), [dragStop](#) and [drop](#) events occur in the mentioned order when a drag and drop operation is performed.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
  $(function () {
    var filePath =
      "http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
    var ajaxActionHandler =
      "http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
    var file = new ej.FileExplorer($("#fileExplorer"), {
      path: filePath,
      ajaxAction: ajaxActionHandler,
      allowDragAndDrop: false
    });
  });
}
```

## Thumbnail Compression

The “FileExplorer” allows thumbnail images to be compressed on the server side and loaded into the “FileExplorer” layout to improve performance when working with large size images.

You can enable this option using “**enableThumbnailCompress**” API of FileExplorer. For enabling this API, [showThumbnail](#) must be set to true in order to render thumbnail preview of images in Tile and LargeIcons layout.

The [beforeGetImage](#) event is triggered before loading a requested image from the server and [getImage](#) event is triggered after the requested image is loaded.

Refer following code block that will be helpful to you.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
var filePath =
"http://js.syncfusion.com/demos/ejServices/webapi/FileExplorer/FileBrowser/"
;
var ajaxActionHandler =
"http://js.syncfusion.com/demos/ejServices/api/fileoperation/doJSONAction";
var file = new ej.FileExplorer($("#fileExplorer"), {
path: filePath,
ajaxAction: ajaxActionHandler,
width: "100%",
layout: "tile",
enableThumbnailCompress: true
});
});
}
```

**Note:** In server side, don't forget to add “[GetImage](#)” handling operation that helps to reduce the image size before loading.

## Accessing shared folder

Using “FileExplorer”, you can manage the files that are available in a shared folder of another system which is connected through LAN. Refer following steps, here you will see the details about accessing the shared folder with FileExplorer.

First you have to specify the shared folder path in any of following formats.

1. “//Server/SharedFolder”
2. “//Server”

Refer following code block to load the shared folder in our FileExplorer component.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
$(function () {
```

```

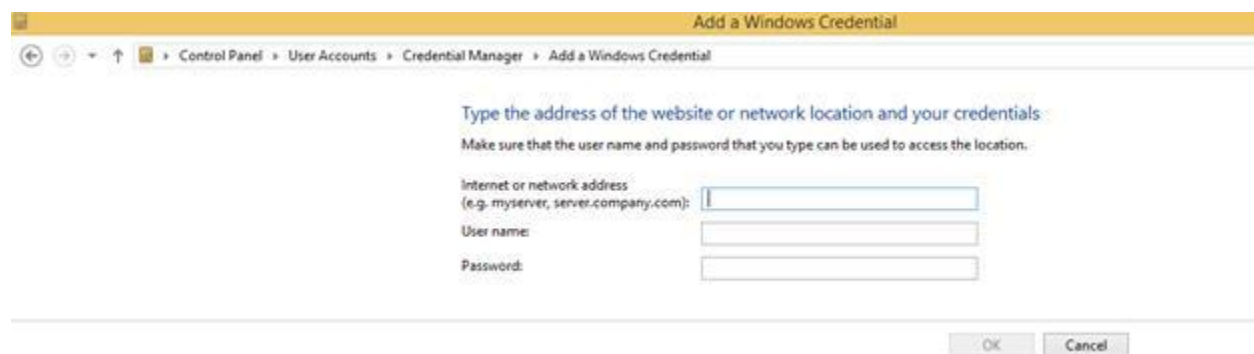
var filePath = "//Server/SharedFolder";
var ajaxActionHandler = "/FileOperations";
var file = new ej.FileExplorer($("#fileExplorer"), {
  path: filePath,
  ajaxAction: ajaxActionHandler,
  width: "100%",
  layout: "tile"
});
});
}

```

**Note:** In above code block, please specify the action method and shared folder path that is integrated with your server.

### Authentication problem in shared folder

If shared folder is restricted with authentication, that time you are not able to access this folder using our FileExplorer. Here you will get an exception like **"System.UnauthorizedAccessException, Please add your windows credential details to open '\\server\'"**. In order to solve this problem, you have to add the shared folder credential details in the windows credential manager that is available in your service hosted machine.



### Localization

The FileExplorer control provides the translations for all static texts and messages. By default the FileExplorer uses the US English (en-US) language, and it having the built-in values of all texts with the corresponding key.

The below table lists out the keys with corresponding texts in English culture.

### JAVASCRIPT

```

ej.FileExplorer.Locale["en-US"] = {
  Back: "Backward",
  Forward: "Forward",
  Upward: "Upward",
  Refresh: "Refresh",
  AddressBar: "Address bar",
  Upload: "Upload",
  Rename: "Rename",
  Delete: "Delete",
  Download: "Download",
  Error: "Error",
  Cut: "Cut",

```



```
Copy: "Copy",
Paste: "Paste",
Details: "Details",
SearchBar: "Search bar",
Open: "Open",
Search: "Search",
EmptyResult: "No items match your search",
EmptyFolder: "This folder is empty",
NewFolder: "New Folder",
Size: "Size",
RenameAlert: "Please enter new name",
NewFolderAlert: "Please enter new folder name",
ContextMenuOpen: "Open",
ContextMenuNewFolder: "New Folder",
ContextMenuDelete: "Delete",
ContextMenuRename: "Rename",
ContextMenuUpload: "Upload",
ContextMenuDownload: "Download",
ContextMenuCut: "Cut",
ContextMenuCopy: "Copy",
ContextMenuPaste: "Paste",
ContextMenuGetinfo: "Get info",
ContextMenuRefresh: "Refresh",
Item: "item",
Items: "items",
ErrorOnFolderCreation: "This destination already contains a folder named '{0}'. Do you want to merge this folder content with already existing folder '{0}'?",
GeneralError: "Please see browser console window for more information",
ErrorPath: "FileExplorer can't find '{0}'. Check the spelling and try again.",
ReplaceAlert: "File named '{0}' already exists. Replace old file with new one?",
DuplicateAlert: "There is already a file with the same name '{0}'. Do you want to create file with duplicate name",
DuplicateFileCreation: "There is already a file with the same name in this location. Do you want to rename '{0}' to '{1}'?",
DeleteFolder: "Are you sure you want to delete ",
DeleteMultipleFolder: "Are you sure you want to delete these {0} items?",
CancelPasteAction: "The destination folder is a subfolder of source folder.",
OkButton: "Ok",
CancelButton: "Cancel",
YesToAllButton: "Yes to all",
NoToAllButton: "No to all",
YesButton: "Yes",
NoButton: "No",
Grid: "Grid View",
Tile: "Tile View",
Name: "Name",
Location: "Location",
Type: "Item type",
Created: "Created",
Accessed: "Accessed",
Modified: "Modified",
DialogCloseToolTip: "close",
UploadSettings: {
```

```
buttonText: {
  upload: "Upload",
  browse: "Browse",
  cancel: "Cancel",
  close: "Close"
},
dialogText: {
  title: "Upload Box",
  name: "Name",
  size: "Size",
  status: "Status"
},
dropAreaText: "Drop files or click to upload",
filedetail: "The selected file size is too large. Please select a file
within the valid size.",
denyError: "Files with #Extension extensions are not allowed.",
allowError: "Only files with #Extension extensions are allowed.",
cancelToolTip: "Cancel",
removeToolTip: "Remove",
retryToolTip: "Retry",
completedToolTip: "Completed",
failedToolTip: "Failed",
closeToolTip: "Close"
}
};
```

### Change the Culture

The language of the FileExplorer can be changed by the [locale](#) property. This property allows the culture code of the country as input. The culture code has to specify in the standard format as **[Language Code]-[County/Region Code]**.

For example the culture code for German is **de-DE**.

You can customize built-in text and messages based on your culture. The below example shows the usage of FileExplorer with German (de-DE) culture.

### JAVASCRIPT

```
ej.FileExplorer.Locale["de-DE"] = {
  Back: "rückwärts",
  Forward: "Nach Vorne",
  Upward: "nach oben",
  Refresh: "erfrischen",
  AddressBar: "Adressleiste",
  Upload: "hochladen",
  Rename: "umbenennen",
  Delete: "löschen",
  Download: "herunterladen",
  Error: "Fehler",
  Cut: "Schnitt",
  Copy: "Kopie",
  Paste: "kleben",
  Details: "Einzelheiten",
  SearchBar: "Search bar",
  Open: "geöffnet",
  Search: "Suche",
```

```
EmptyResult: "Keine Artikel entsprechen Ihrer Suche nach",
EmptyFolder: "Dieser Ordner ist leer",
NewFolder: "neuer Ordner",
SelectedFileUrl: "Web-Adresse",
SelectedFileName: "Titel",
ImageWidth: "Breite",
ImageHeight: "Höhe",
Insert: "Insert",
Cancel: "Rückgängig Machen",
RenameAlert: "Bitte geben Sie neuen Namen",
NewFolderAlert: "Geben Sie den neuen Ordnernamen ein",
ContextMenuOpen: "geöffnet",
ContextMenuNewFolder: "Neuer Ordner",
ContextMenuDelete: "löschen",
ContextMenuRename: "umbenennen",
ContextMenuUpload: "hochladen",
ContextMenuDownload: "Herunterladen",
ContextMenuCut: "Schnitt",
ContextMenuCopy: "Kopie",
ContextMenuPaste: "kleben",
ContextMenuGetinfo: "Ausführliche Infos",
ContextMenuRefresh: "erfrischen",
Item: " Artikel",
Items: " Angebote",
OkButton: "Ok",
CancelButton: "Rückgängig machen",
YesToAllButton: "Ja zu allem",
NoToAllButton: "Nein, alle",
YesButton: "Ja",
NoButton: "Unterlassen Sie",
Size: "Größe",
Grid: "Gitter Ansicht",
Tile: "Kachelansicht",
ErrorOnFolderCreation: "Dieses Ziel ist bereits ein Ordner mit dem Namen '{0}'. Möchten Sie diesen Ordner Inhalte mit bereits vorhandenen Ordner '{0}' zusammenführen möchten?",
GeneralError: "Bitte beachten Sie Browser Konsolenfenster für weitere Informationen",
ErrorPath: "Fileexplorer kann nicht finden '{0}'. Überprüfen Sie die Schreibweise und versuchen Sie es erneut.",
ReplaceAlert: "Datei mit dem Namen '{0}' ist bereits vorhanden. Ersetzen Sie alte Datei durch eine neue?",
DuplicateAlert: "Es gibt bereits eine Datei mit dem gleichen Namen '{0}'. Möchten Sie diese Datei mit doppelten Namen erstellen möchten",
DuplicateFileCreation: "Es gibt bereits eine Datei mit dem gleichen Namen in diesem Ort. Möchten Sie umbenennen '{0}' bis '{1}' suchen?",
DeleteFolder: " Sind Sie sicher, dass Sie löschen möchten ",
DeleteMultipleFolder: "Sind Sie sicher, dass Sie diese {0} Einträge löschen?",
CancelPasteAction: "Der Zielordner ist ein Unterordner des Quellordners.",
Name: "Namen",
Location: "Ort",
Type: "Gegenstandsart",
Created: "Erstellt",
Modified: "geändert",
DialogCloseToolTip: "schließen",
UploadSettings: {
```

```

buttonText: {
  upload: "hochladen",
  browse: "blättern",
  cancel: "Rückgängig Machen",
  close: "schließen"
},
dialogText: {
  title: "hochladen Box",
  name: "Name",
  size: "Größe",
  status: "Status"
},
cancelToolTip: "stornieren",
removeToolTip: "entfernen",
retryToolTip: "wiederholen",
completedToolTip: "fertiggestellt",
failedToolTip: "fehlgeschlagen",
closeToolTip: "schließen"
}
};

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ExplorerComponent {
  $(function () {
    var filePath =
      "http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/";
    var ajaxActionHandler =
      "http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOperations";
    var file = new ej.FileExplorer($("#fileExplorer"), {
      path: filePath,
      ajaxAction: ajaxActionHandler,
      // it sets the localization value as "de-DE"
      // and it gets the text and message values of this culture from the above //
      definition: ej.FileExplorer.Locale["de-DE"]
      locale: "de-DE"
    });
  });
}

```

## Keyboard Navigation

The FileExplorer control can be fully accessible with the help of keyboard shortcuts and it can be allowed by specifying [allowKeyboardNavigation](#) as true. The below table shows the list of keyboard shortcuts to access the FileExplorer control.

**Note:** set focus on FileExplorer component to access the keyboard shortcuts.

| Key              | Description         |
|------------------|---------------------|
| Ctrl + Shift + 1 | Focuses the Toolbar |

|                  |                                  |
|------------------|----------------------------------|
| Ctrl + Shift + 2 | Focuses the Tree view            |
| Ctrl + Shift + 3 | Focuses the Splitter             |
| Ctrl + Shift + 4 | Focuses the Grid view            |
| Ctrl + Shift + 5 | Focuses the Thumbnail view       |
| Ctrl + Shift + 6 | Focuses the Status bar           |
| Right            | Focusing next item               |
| Left             | Focusing previous item           |
| Enter            | Selection the focused item       |
| Esc              | Closes the opened dialog         |
| F2               | Renames the selected item        |
| Ctrl + A         | Selects the all items            |
| Alt + N          | Opens Create New Folder dialog   |
| Ctrl + U         | Opens Upload dialog              |
| F5               | Refresh the content              |
| Alt + Enter      | Get the details of selected item |
| Ctrl + X         | To cut the selected items        |
| Ctrl + C         | To copy the selected items       |
| Ctrl + V         | Pastes the copied items          |
| Delete           | Deletes the selected items       |
| Shift + F10      | Opens the Context Menu           |

## Gantt

### Overview

The Essential TypeScript Gantt control is designed to visualize and edit the project schedule, and track the project progress. It helps to organize and schedule the projects and also you can update the project schedule through interactions like editing, dragging and resizing.

### Key Features

- Sorting
- Editing
- Task relationship
- Stripline
- Baseline
- Timeline customization
- Work break down structure
- Taskbar customization
- Task label customization
- Localization
- Resources

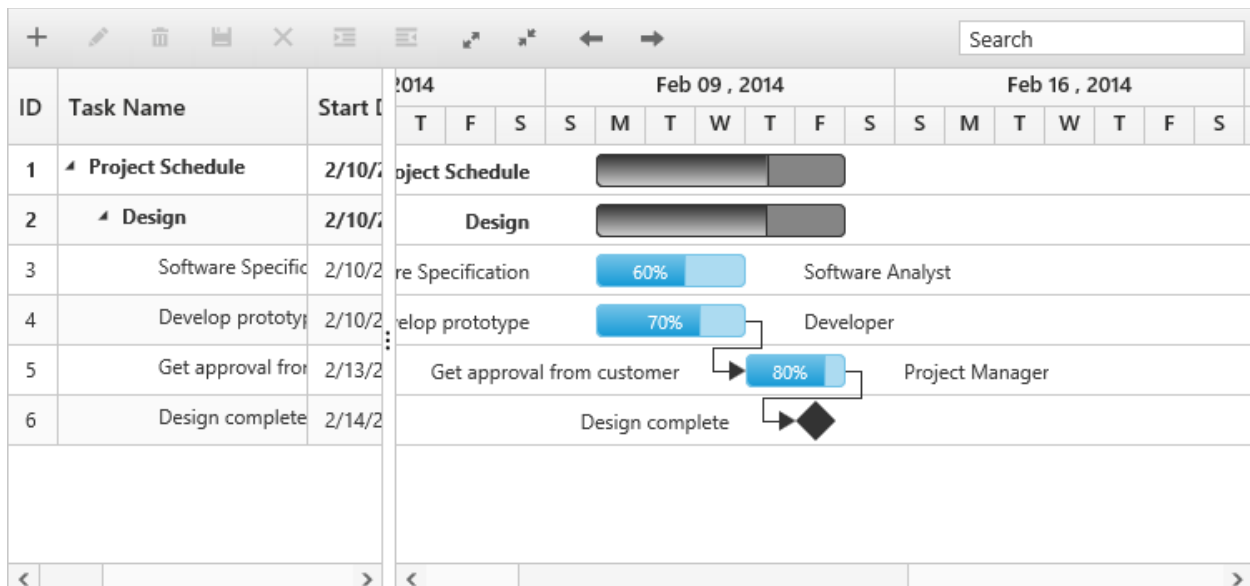
### Getting Started

This section explains briefly about how to create a Gantt chart in your application with TypeScript.

#### Create your first Gantt in TypeScript

To get started Syncfusion TypeScript application refer [this](#) page for basic control integration and script references.

In this tutorial, you can learn how to create a simple Gantt chart, add tasks or subtasks, and set relationship between tasks during the design phase of a software project. The following screenshot displays the desired output after completing this tutorial,



It is necessary to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the jquery.d.ts file in your TypeScript application, which can be downloaded from [here](#).

### Adding script references

Create an HTML file and add the following template to the HTML file.

To get started, you can use the ej.web.all.min.js file that encapsulates all the ej controls and frameworks in one single file. So the complete boilerplate code is

### HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Getting Started with Gantt Control for Aurelia</title>
<!-- style sheet for default theme(flat azure) -->
<link href="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />
<!--scripts-->
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-1.10.2.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js "></script>
</head>
<body>
<!--Add Gantt control here-->
</body>
</html>
```

### Initialize the Gantt

Add a <div> element with in the <Body> tag.

### HTML

```
<div class="cols-sample-area">
<div id="GanttContainer"></div>
</div>
<script type="text/javascript" src="gant/gantt.js"></script>
```

### JAVASCRIPT

```
projectData = [
{
taskID:1,
taskName: "Project Schedule",
startDate: "02/03/2014",
```

```
endDate: "03/07/2014",
subtasks: [
{
taskID:2,
taskName: "Planning",
startDate: "02/03/2014",
endDate: "02/07/2014"
},
]
//...
}
];
```

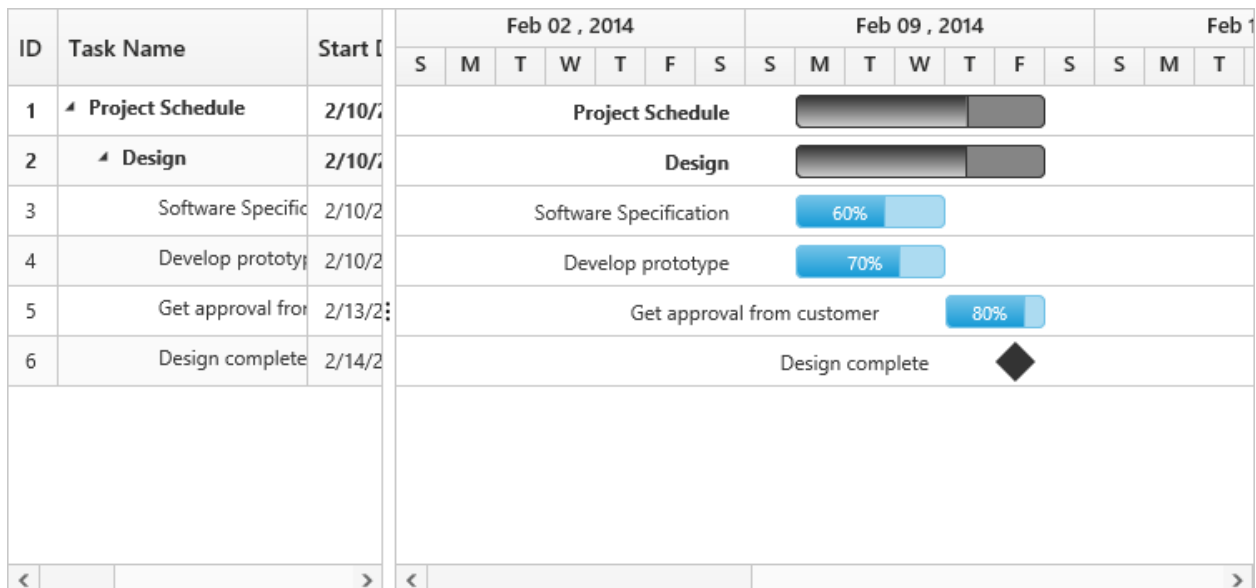
Initialize the Gantt with data source created in the last step.

### JS

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module GanttComponent {
$(function() {
var ganttInstance = new ej.Gantt($("#GanttContainer"), {
dataSource: projectData,
taskIdMapping: "taskID",
allowDragAndDrop: true,
taskNameMapping: "taskName",
startDateMapping: "startDate",
progressMapping: "progress",
durationMapping: "duration",
endDateMapping: "endDate",
childMapping: "subtasks"
//...
});
});
}
```

A Gantt chart is created as shown in the following screen shot.





### Enable Toolbar

Gantt control contains toolbar options to edit, search, expand or collapse all records, indent, outdent, delete, and add a task. You can enable toolbar using the [toolbarSettings](#) property.

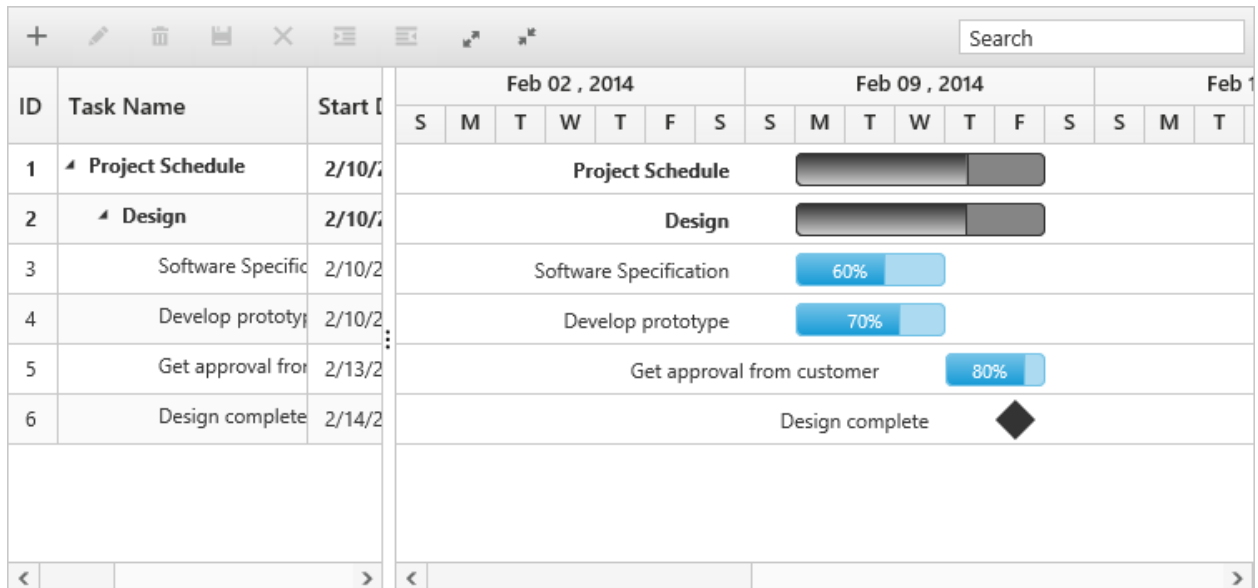
### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module GanttComponent {
  $(function() {
    var ganttInstance = new ej.Gantt($("#GanttContainer"), {
      toolbarSettings: {
        showToolbar: true,
        toolbarItems: ["add", "edit", "delete", "update", "cancel", "indent",
          "outdent", "expandAll", "collapseAll", "search"]
      },
      //...
    });
  });
}

```

The following screen shot displays a Tool bar in Gantt chart control:



**Note:** Add, edit, delete, indent and outdent options are enabled when enabling the `allowEditing`, `allowAdding`, `allowDelete`, `allowIndent` and `allowOutdent` properties in the edit Options.

### Enable Sorting

The Gantt control has sorting functionality to arrange the tasks in ascending or descending order based on a particular column.

#### Multicolumn Sorting

Enable the multicolumn sorting in Gantt by setting `allowMultiSorting` as `true`. You can sort multiple columns in Gantt, by selecting the desired column header while holding the **CTRL** key.

### JAVASCRIPT

```

/// <reference path="../../../tsfiles/jquery.d.ts" />
/// <reference path="../../../tsfiles/ej.web.all.d.ts" />
module GanttComponent {
    $(function() {
        var ganttInstance = new ej.Gantt($("#GanttContainer"), {
            allowSorting: true,
            allowMultiSorting: true
            //...
        });
    });
}

```

### Enable Editing

You can enable editing using `editSettings` and `allowGanttChartEditing` options.

#### Cell Editing

Modify the task details through the grid cell editing by setting the `editMode` as `cellEditing`.

#### Normal Editing

Modify the task details through the edit dialog by setting the `editMode` as `normal`.

#### Taskbar Editing

Modify the task details through user interaction such as resizing and dragging the taskbar.

*Predecessor Editing*

Modify the predecessor details of a task using mouse interactions by setting [allowGanttChartEditing](#) as true and setting the value for predecessorMapping property.

**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module GanttComponent {
$(function() {
var ganttInstance = new ej.Gantt($("#GanttContainer"), {
allowGanttChartEditing: true,
predecessorsMapping: "predecessor",
editSettings: {
allowEditing: true,
allowAdding: true,
allowDeleting: true,
allowIndent: true,
editMode: "cellEditing"
},
//...
});
});
}

```

The following screen shot displays a Gantt chart control with Enable Editing options.

The screenshot shows a 'Task Information' dialog box with a close button (X) in the top right corner. The dialog has four tabs: 'General' (selected), 'Predecessors', 'Resources', and 'Custom Fields'. The 'General' tab contains the following fields:

- ID:** A text box containing the value '3'.
- Task Name:** An empty text box.
- Start Date:** A date picker showing '2/3/2014'.
- End Date:** A date picker showing '2/7/2014'.
- Duration:** A text box showing '5 days'.
- Progress:** A slider control set to '100'.
- Work:** A slider control set to '40'.
- Task Type:** A dropdown menu showing 'Fixed Units'.
- Effort Driven:** A dropdown menu showing 'No'.

At the bottom of the dialog, there are three buttons: 'Delete Task', 'Save', and 'Cancel'.

**Note:** Both cellEditing and normal editing operations are performed through double-click action.

### Enable Context Menu

You can enable the context menu in Gantt, by setting the [enableContextMenu](#) as `true`.

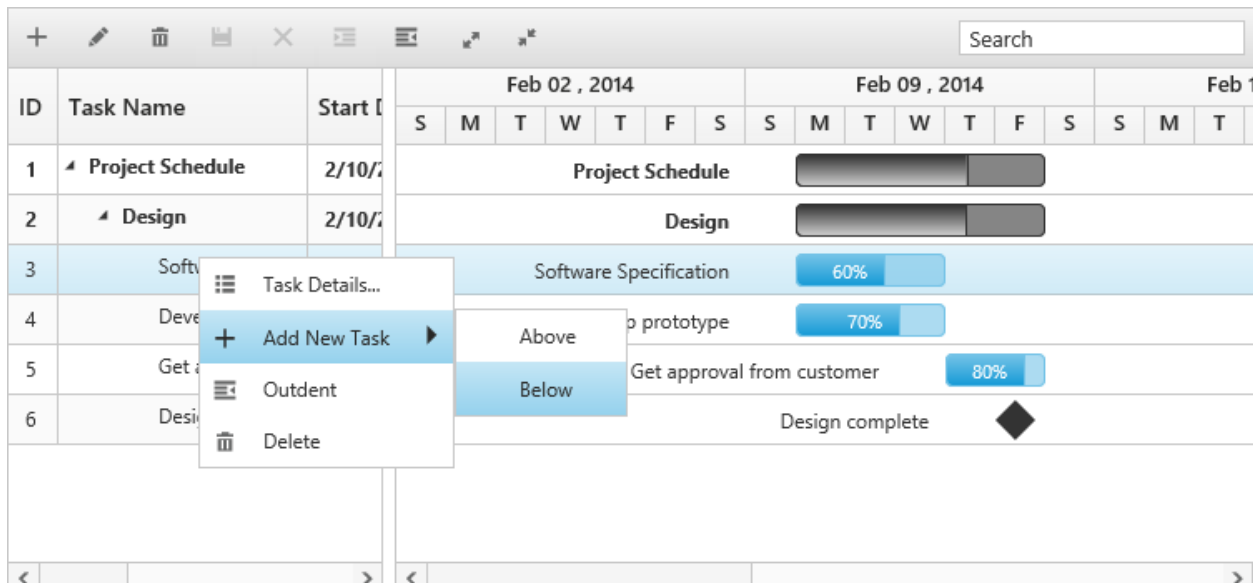
#### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module GanttComponent {
$(function() {
var ganttInstance = new ej.Gantt($("#GanttContainer"), {
enableContextMenu: true,
//...
});
});
}

```

The following screen shot displays Gantt chart in which Context menu option is enabled:



### Enable Column Menu

You can enable the column menu in Gantt, by setting the [showColumnChooser](#) as `true`.

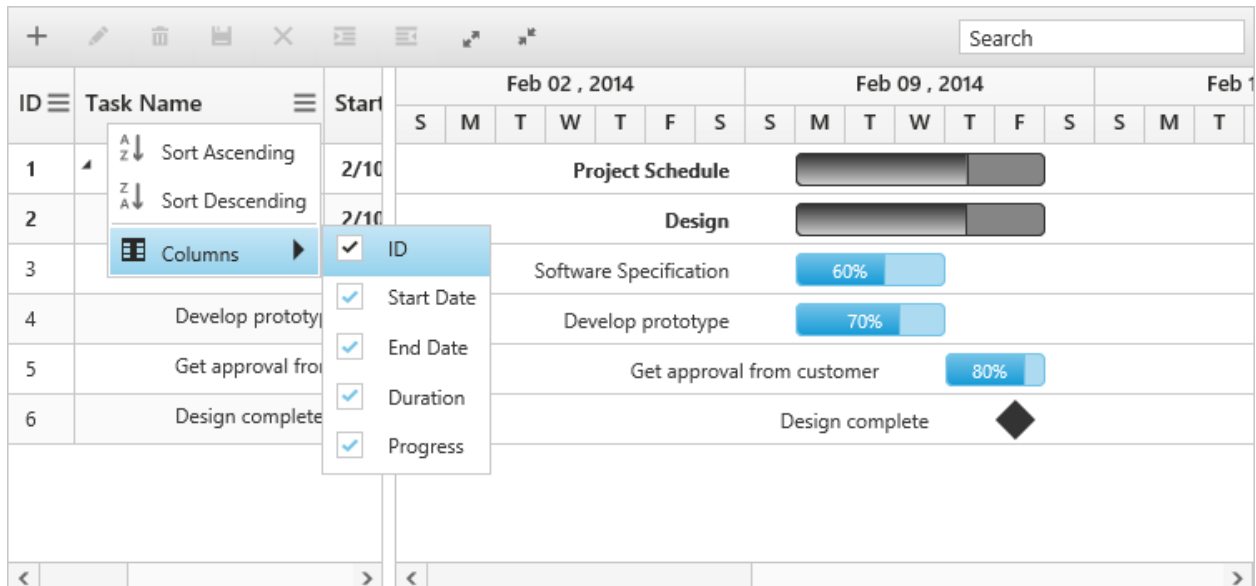
#### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module GanttComponent {
$(function() {
var ganttInstance = new ej.Gantt($("#GanttContainer"), {
showColumnChooser: true,
//...
});
});
}

```

The following screen shot displays Gantt chart in which column chooser option is enabled:



### Provide tasks relationship

In Gantt, you have the predecessor support to show the relationship between two different tasks.

- **Start to Start (SS)** - You cannot start a task until the other task also starts.
- **Start to Finish (SF)** - You cannot finish a task until the other task finishes.
- **Finish to Start (FS)** - You cannot start a task until the other task completes.
- **Finish to Finish (FF)** - You cannot finish a task until the other task completes.

You can show the relationship in tasks, by using the [predecessorMapping](#), as shown in the following code example.

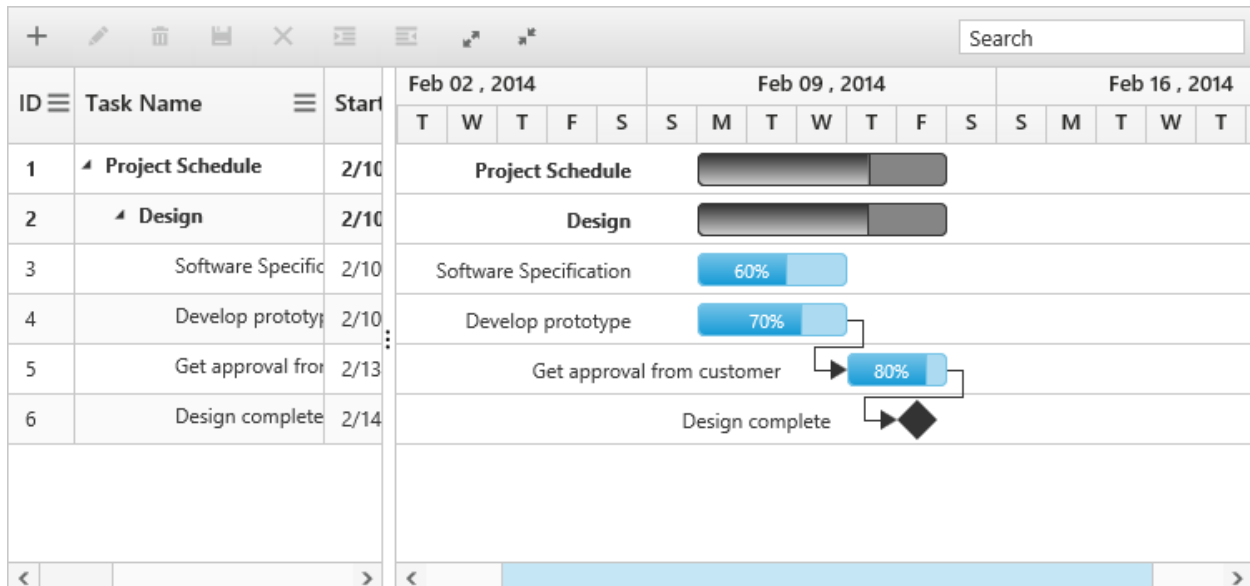
### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module GanttComponent {
$(function() {
var ganttInstance = new ej.Gantt($("#GanttContainer"), {
predecessorsMapping: "predecessor",
//...
});
});
}

```

The following screenshot displays the relationship between tasks.



### Provide Resources

In Gantt control, you can display and assign the resource for each task. Create a collection of **JSON** object, which contains id and name of the resource and assign it to [resources](#) property. Then, specify the field name for id and name of the resource in the resource collection to [resourceIdMapping](#) and [resourceNameMapping](#) options. The name of the field, which contains the actual resources assigned for a particular task in the **dataSource** is specified using [resourceInfoMapping](#).

1. Create the resource collection to be displayed in ejGantt

#### JAVASCRIPT

```
projectResources = [{
  resourceId: 1,
  resourceName: "Project Manager"
}, {
  resourceId: 2,
  resourceName: "Software Analyst"
}, {
  resourceId: 3,
  resourceName: "Developer"
}, {
  resourceId: 4,
  resourceName: "Testing Engineer"
}];
```

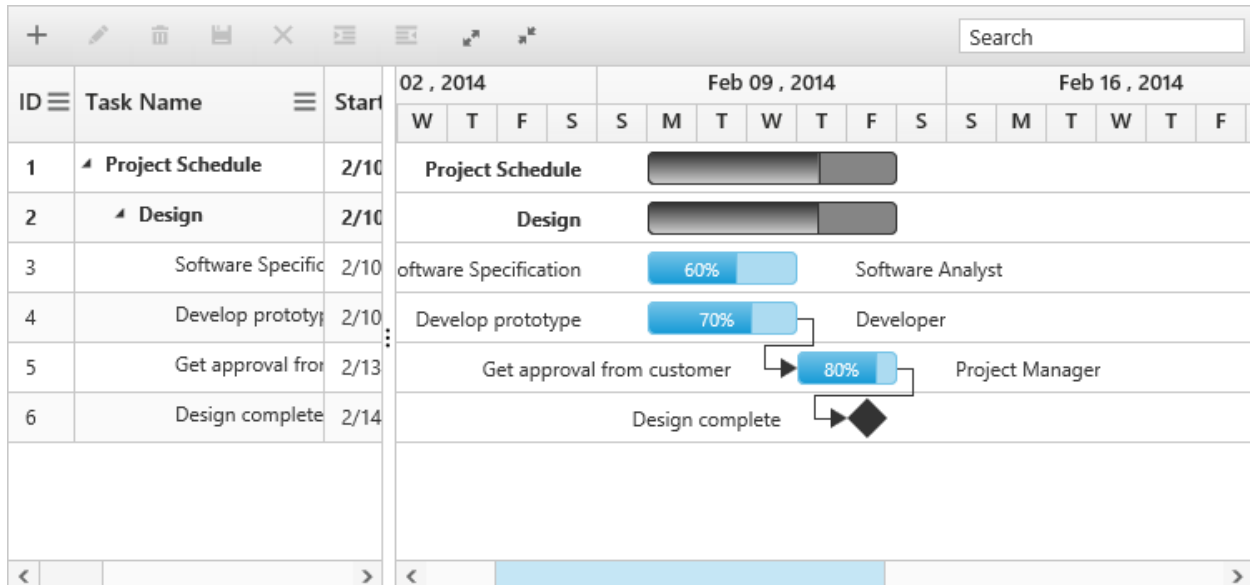
2. Initialize the Gantt with resources created in last step.

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module GanttComponent {
  $(function() {
    var ganttInstance = new ej.Gantt($("#GanttContainer"), {
      resourceInfoMapping: "resourceId",
      resourceNameMapping: "resourceName",
    });
  });
}
```

```
resourceIdMapping: "resourceId",
resources: projectResources,
//...
});
});
}
```

The following screenshot displays resource allocation for tasks in Gantt chart.



By following these steps, you have learned how to provide data source to Gantt chart, how to configure Gantt to set task relationships, assign resources for each task, and add toolbar with necessary buttons.

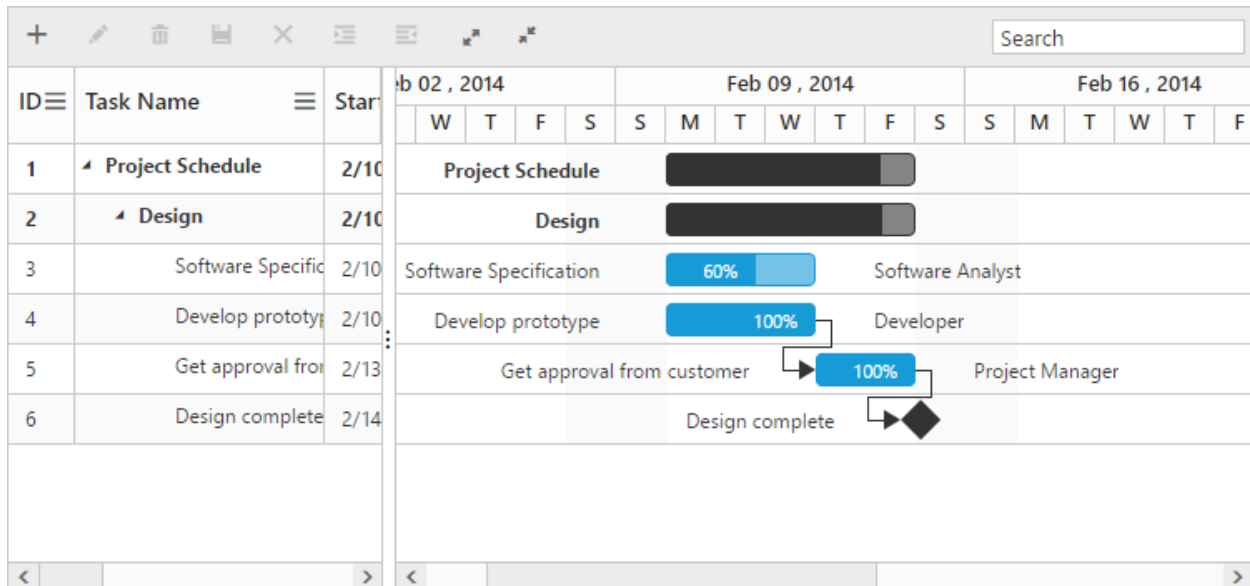
### Highlight Weekend

In Gantt, you can on or off weekends high lighting by setting the [highlightWeekends](#) as `true` or `false`.

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module GanttComponent {
$(function() {
var ganttInstance = new ej.Gantt($("#GanttContainer"), {
highlightWeekends = false
//...
});
});
}
```

The following screen shot displays Gantt chart in which highlight weekends is disabled:



## Grid

### Overview

The Grid control for TypeScript is an efficient display engine for tabular data. It will pull data from a datasource, such as array of JSON objects, OData web services, or ej.DataManager; binding data fields to columns and displaying a column header to identify the field. It is a feature-rich control that provides extensive appearance customization options with support for grouped records. This Grid is very useful for generating complex grid-based reports with rich formatting. The most important features available in the Grid control for TypeScript are paging, sorting, filtering, searching, grouping, and editing.

### Key Features

Some important features of the Grid control are:

- **Datasources** - Bind the Grid control with an array of JSON objects or ej.DataManager.
- **Responsive** - Provides three level of responsive modes which includes redesigned UI for Mobile.
- **Sorting and grouping** - Supports n levels of sorting and grouping.
- **Filtering** - Offers Excel-like filtering for filter data.
- **Editing** - Offers two editing modes for inserting, editing, and deleting records in a grid.
- **Paging** - Provides the option to easily switch between pages using the pager bar.
- **Reordering** - Allows you to drag any column and drop it at any position in the Grid's column header row, allowing columns to be repositioned at the required place.
- **Resize columns** - Grid provides option for resizing the columns.
- **Summary support** - Offers options for specifying summary rows and columns.
- **Detail template** - Offers to render the detail row for the corresponding expanded master row.
- **Unbound columns** - Offers the option to specify unbound columns.
- **RTL support** - Provides a full-fledged right-to-left mode which aligns content in the **Grid** control from right to left.
- **Localization** - Provides inherent support to localize the UI.



## Getting started

Before we start with the Grid, for common getting started of TypeScript please refer [this page](#) provides general information regarding integrating Syncfusion widget's.

To render the grid control in TypeScript platform, the important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (grid.ts file), so that you will get the intellisense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the jquery.d.ts file in your TypeScript application, which can be downloaded from [here](#).

In your TypeScript app folder, create "grid.ts" file and now refer these two files within the grid.ts file as shown below,

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

## Script and CSS Reference

Add the scripts and CSS references to the "index.html" page as the order mentioned below and also define the Grid control.

The Grid control has the following list of external JavaScript dependencies.

- [jQuery](#) 1.7.1 and later versions
- [jsRender](#) - to render the templates.

Refer to the internal dependencies in the following table.

| File                   | Description/Usage  |
|------------------------|--|
| ej.core.min.js         | It is referred always before using all the JS controls.  |
| ej.data.min.js         | Used to handle data operation and is used while binding data to the JS controls.   |
| ej.touch.min.js        | Used to handle touch operations in touch-enabled devices   |
| ej.print.min.js        | Used to handle print operation in JS controls.   |
| ej.globalize.min.js    | It is referred when using localization in Grid.  |
| ej.draggable.min.js    | Used for drag and drop an element in JS controls.  |
| ej.grid.min.js         | The grid's main file.  |
| ej.pager.min.js        | It is referred when paging is used in the Grid.  |
| ej.scroller.min.js     | It is referred when scrolling is used in the Grid.   |
| ej.waitingpopup.min.js | It is referred when the remote data binding is used in the Grid. The waiting popup shows while requesting the server for data. |

|                             |  |
|-----------------------------|--|
| ej.dropdownlist.min.js      | These files are used while enable the Editing and Filtering feature in the Grid. |
| ej.dialog.min.js            |  |
| ej.button.min.js            |  |
| ej.autocomplete.min.js      |  |
| ej.datepicker.min.js        |  |
| ej.datetimestepicker.min.js |  |
| ej.timepicker.min.js        |  |
| ej.checkbox.min.js          |  |
| ej.editor.min.js            |  |
| ej.tooltip.js               | It is referred when toolbar is enabled in Grid.                                  |
| ej.toolbar.min.js           |  |
| ej.menu.js                  | It is referred when excel like filter menu or context menu is enabled.           |
| ej.radiobutton.js           | It is referred when filtering is enabled.  |
| ej.excelfilter.js           | It is referred when excel like filter menu is enabled.                           |

To get started, you can use the `ej.web.all.min.js` file that encapsulates all the `ej` controls and frameworks in one single file. So the complete boilerplate code is

### HTML

```

<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential Studio for JavaScript">
<meta name="author" content="Syncfusion">
<title></title>
<!-- Essential Studio for JavaScript theme reference -->
<link rel="stylesheet"
href="http://cdn.syncfusion.com/14.4.0.15/js/web/flat-
azure/ej.web.all.min.css" />
<!-- Essential Studio for JavaScript script references -->
<script src="https://code.jquery.com/jquery-1.10.2.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"></script>
<script src="http://cdn.syncfusion.com/14.4.0.15/js/web/ej.web.all.min.js">
</script>
<!-- Add your custom scripts here -->
<script type="text/javascript" src="grid.js"></script>      <!--Also refer
grid.js file here -->
</head>
<body>
<div id="Grid"></div>      <!-- Define the Grid control-->

```

```
</body>
</html>
```

**Note:** In production, we highly recommend you to use our [custom script generator](#) to create custom script file with required controls and its dependencies only. Also to reduce the file size further please use [GZip compression](#) in your server.

For themes, you can use the `ej.web.all.min.css` CDN link from the code example given. To add the themes in your application, please refer to [this link](#). In addition to that, refer the `grid.js` file in the script section.

Finally build your application, so that the “`grid.js`” file is automatically generated and got added to your project (User have nothing to do with this file). Now, whatever code changes that you make in `grid.ts` file will be reflected in `grid.js` file automatically.

### Create a Grid

The grid can be created from a HTML DIV element with the HTML `id` attribute set to it and define these steps in “`index.html`” page.

#### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="Grid"></div>
<script>
var shipDetails = [
{ Name: 'Hanari Carnes', City: 'Brazil' },
{ Name: 'Split Rail Beer & Ale', City: 'USA' },
{ Name: 'Ricardo Adocicados', City: 'Brazil' }
];
</script>
</body>
</html>
```

Initialize the Grid in `grid.ts` file by using the `ej.Grid` method. Refer to the following code example.

#### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
dataSource: shipDetails,
});
});
}
```

| Name                  | City   |
|-----------------------|--------|
| Hanari Carnes         | Brazil |
| Split Rail Beer & Ale | USA    |
| Ricardo Adocicados    | Brazil |

### Data Binding

[Data binding](#) in the grid is achieved by assigning an array of JavaScript objects to the [dataSource](#) property. Refer to the following code example.

### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="Grid"></div>
</body>
</html>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window['gridData']" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"];
columns: ["OrderID", "EmployeeID", "CustomerID", "ShipCountry", "Freight"];
});
});
}
```

| OrderID | EmployeeID | CustomerID | ShipCountry | Freight |
|---------|------------|------------|-------------|---------|
| 10248   | 5          | VINET      | France      | 32.38   |
| 10249   | 6          | TOMSP      | Germany     | 11.61   |
| 10250   | 4          | HANAR      | Brazil      | 65.83   |
| 10251   | 3          | VICTE      | France      | 41.34   |
| 10252   | 4          | SUPRD      | Belgium     | 51.3    |
| 10253   | 3          | HANAR      | Brazil      | 58.17   |
| 10254   | 5          | CHOPS      | Switzerland | 22.98   |
| 10255   | 9          | RICSU      | Switzerland | 148.33  |

### Enable Paging

[Paging](#) can be enabled by setting the [allowPaging](#) to true. The Paging feature in Grid offers complete navigation support to easily switch between the pages, using the page bar available at the bottom of the Grid control

### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="Grid"></div>
</body>
</html>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window['gridData']" is referred from
            // 'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            columns : ["OrderID", "EmployeeID", "CustomerID", "ShipCountry", "Freight"],
            allowPaging: true,
            pageSettings: { pageSize: 8 }
        });
    });
}
```

| OrderID | EmployeeID | CustomerID | ShipCountry | Freight |
|---------|------------|------------|-------------|---------|
| 10248   | 5          | VINET      | France      | 32.38   |
| 10249   | 6          | TOMSP      | Germany     | 11.61   |
| 10250   | 4          | HANAR      | Brazil      | 65.83   |
| 10251   | 3          | VICTE      | France      | 41.34   |
| 10252   | 4          | SUPRD      | Belgium     | 51.3    |
| 10253   | 3          | HANAR      | Brazil      | 58.17   |
| 10254   | 5          | CHOPS      | Switzerland | 22.98   |
| 10255   | 9          | RICSU      | Switzerland | 148.33  |

1
2
3
4
5
6
7
8
...

1 of 25 pages (200 items)

**Note:** Pager settings can be customized by using the `pageSize` of [pageSettings](#) property. When it is not given the default values for `pageSize` and `pageCount` are 12 and 8 respectively.

### Enable Filtering

[Filtering](#) can be enabled by setting the `[allowFiltering]`

(<https://help.syncfusion.com/js/api/ejgrid#members:allowfiltering>) to be `true`. By default, the filter bar row is displayed to perform filtering, you can change the filter type by using `filterType` of [filterSetting](#) property.

### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="Grid"></div>
</body>
</html>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window['gridData'] is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
columns : ["OrderID", "EmployeeID", "CustomerID", "ShipCountry", "Freight"],
allowPaging: true,
pageSettings: { pageSize: 8 },
allowFiltering: true
});
});
}
```

```
});
});
}
```

| OrderID | EmployeeID | CustomerID | ShipCountry | Freight |
|---------|------------|------------|-------------|---------|
|         |            |            |             |         |
| 10248   | 5          | VINET      | France      | 32.38   |
| 10249   | 6          | TOMSP      | Germany     | 11.61   |
| 10250   | 4          | HANAR      | Brazil      | 65.83   |
| 10251   | 3          | VICTE      | France      | 41.34   |
| 10252   | 4          | SUPRD      | Belgium     | 51.3    |
| 10253   | 3          | HANAR      | Brazil      | 58.17   |
| 10254   | 5          | CHOPS      | Switzerland | 22.98   |
| 10255   | 9          | RICSU      | Switzerland | 148.33  |

1
2
3
4
5
6
7
8
...

1 of 25 pages (200 items)

### Enable Grouping

[Grouping](#) can be enabled by setting the [allowGrouping](#) to `true`. Columns can be grouped dynamically by drag and drop the grid column header to the group drop area. The initial grouping can be done by adding required column names in the [groupedColumns](#) of [groupSettings](#) property.

### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="Grid"></div>
</body>
</html>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window['gridData']" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
columns : ["OrderID", "EmployeeID", "CustomerID", "ShipCountry", "Freight"],
```

```

allowPaging: true,
pageSettings: { pageSize: 8 },
allowGrouping: true
});
});
}

```

| OrderID | EmployeeID | CustomerID | ShipCountry | Freight |
|---------|------------|------------|-------------|---------|
| 10248   | 5          | VINET      | France      | 32.38   |
| 10249   | 6          | TOMSP      | Germany     | 11.61   |
| 10250   | 4          | HANAR      | Brazil      | 65.83   |
| 10251   | 3          | VICTE      | France      | 41.34   |
| 10252   | 4          | SUPRD      | Belgium     | 51.3    |
| 10253   | 3          | HANAR      | Brazil      | 58.17   |
| 10254   | 5          | CHOPS      | Switzerland | 22.98   |
| 10255   | 9          | RICSU      | Switzerland | 148.33  |

1 of 25 pages (200 items)

Refer to the following code example for initial grouping.

### HTML

```

<!DOCTYPE html>
<html>
<body>
<div id="Grid"></div>
</body>
</html>

```

### TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window['gridData'] is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
columns : ["OrderID", "EmployeeID", "CustomerID", "ShipCountry", "Freight"],
allowPaging: true,
pageSettings: { pageSize: 8 },

```



```
allowGrouping: true,
groupSettings: { groupedColumns: ["ShipCountry", "CustomerID"] }
});
});
}
```

ShipCountry ^ >

CustomerID ^

| OrderID                         | EmployeeID | CustomerID ^ | ShipCountry ^ | Freight |
|---------------------------------|------------|--------------|---------------|---------|
| ShipCountry: Argentina - 1 item |            |              |               |         |
| CustomerID: OCEAN - 1 item      |            |              |               |         |
| 10409                           | 3          | OCEAN        | Argentina     | 29.83   |
| ShipCountry: Austria - 2 items  |            |              |               |         |
| CustomerID: ERNSH - 10 items    |            |              |               |         |
| 10258                           | 1          | ERNSH        | Austria       | 140.51  |
| 10263                           | 9          | ERNSH        | Austria       | 146.06  |
| 10351                           | 1          | ERNSH        | Austria       | 162.33  |
| 10368                           | 2          | ERNSH        | Austria       | 101.95  |
| 10382                           | 4          | ERNSH        | Austria       | 94.77   |
| 10390                           | 6          | ERNSH        | Austria       | 126.38  |
| 10402                           | 8          | ERNSH        | Austria       | 67.88   |

⏪

⏩

1

2

3

4

5

6

7

8

...

⏪

⏩

1 of 25 pages (200 items)

### Add Summaries

[Summaries](#) can be added by setting the [showSummary](#) to true and adding required summary rows and columns in the [summaryRows](#) property. For demonstration, Freight column's sum value is displayed as summary.

### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="Grid"></div>
</body>
</html>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window['gridData']" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
columns : ["OrderID", "EmployeeID", "CustomerID", "ShipCountry", "Freight"],
allowPaging: true,
pageSettings: { pageSize: 8 },
allowGrouping: true,
groupSettings: { groupedColumns: ["CustomerID"] },
showSummary: true,
summaryRows: [
{
title: "Sum",
summaryColumns: [
{ summaryType: ej.Grid.SummaryType.Sum, displayColumn: "Freight",
dataMember: "Freight" }
]
}
]
});
});
}
```

CustomerID ^ X

| OrderID                     | EmployeeID | CustomerID ^ | ShipCountry | Freight   |
|-----------------------------|------------|--------------|-------------|-----------|
| CustomerID: ANATR - 1 item  |            |              |             |           |
| 10308                       | 7          | ANATR        | Mexico      | 1.61      |
| Sum                         |            |              |             | 1.61      |
| CustomerID: ANTON - 1 item  |            |              |             |           |
| 10365                       | 3          | ANTON        | Mexico      | 22        |
| Sum                         |            |              |             | 22.00     |
| CustomerID: AROUT - 2 items |            |              |             |           |
| 10355                       | 6          | AROUT        | UK          | 41.95     |
| 10383                       | 8          | AROUT        | UK          | 34.24     |
| Sum                         |            |              |             | 76.19     |
| CustomerID: BERGS - 5 items |            |              |             |           |
| 10278                       | 8          | BERGS        | Sweden      | 92.69     |
| 10280                       | 2          | BERGS        | Sweden      | 8.98      |
| 10384                       | 3          | BERGS        | Sweden      | 168.64    |
| 10444                       | 3          | BERGS        | Sweden      | 3.5       |
| Sum                         |            |              |             | 283.11    |
| Sum                         |            |              |             | 13,126.33 |

1

2

3

4

5

6

7

8

...

▶

⏭

1 of 25 pages (200 items)

### Data binding

The Grid control uses [ej.DataManager](#) which supports both RESTful JSON data services binding and local JSON array binding. The [dataSource](#) property can be assigned either with the instance of [ej.DataManger](#) or JSON data array collection. It supports different kinds of data binding methods such as

1. Local data
2. Remote data

### Local Data

To bind local data to the Grid, you can assign a JSON array to the [dataSource](#) property.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window['gridData']" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            allowPaging: true,
            columns: ["OrderID", "EmployeeID", "ShipCity", "ShipCountry", "Freight"]
        });
    });
}
```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID | ShipCity       | ShipCountry | Freight |
|---------|------------|----------------|-------------|---------|
| 10248   | 5          | Reims          | France      | 32.38   |
| 10249   | 6          | Münster        | Germany     | 11.61   |
| 10250   | 4          | Rio de Janeiro | Brazil      | 65.83   |
| 10251   | 3          | Lyon           | France      | 41.34   |
| 10252   | 4          | Charleroi      | Belgium     | 51.3    |
| 10253   | 3          | Rio de Janeiro | Brazil      | 58.17   |
| 10254   | 5          | Bern           | Switzerland | 22.98   |
| 10255   | 9          | Genève         | Switzerland | 148.33  |
| 10256   | 3          | Resende        | Brazil      | 13.97   |
| 10257   | 4          | San Cristóbal  | Venezuela   | 81.91   |
| 10258   | 1          | Graz           | Austria     | 140.51  |
| 10259   | 4          | México D.F.    | Mexico      | 3.25    |

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

**Note:** 1. There is no in-built support to bind the XML data to the grid. But you can achieve this requirement with the help of [custom adaptor](#) concept.

2. Refer this [Knowledge Base link](#) for bounding XML data to grid using custom adaptor.

### Remote Data

To bind remote data to Grid Control, you can assign a service data as an instance of [ej.DataManager](#) to the [dataSource](#) property.

### OData

OData is a standardized protocol for creating and consuming data. You can provide the [OData service](#) URL directly to the [ej.DataManager](#) class and then you can assign it to Grid [dataSource](#).

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

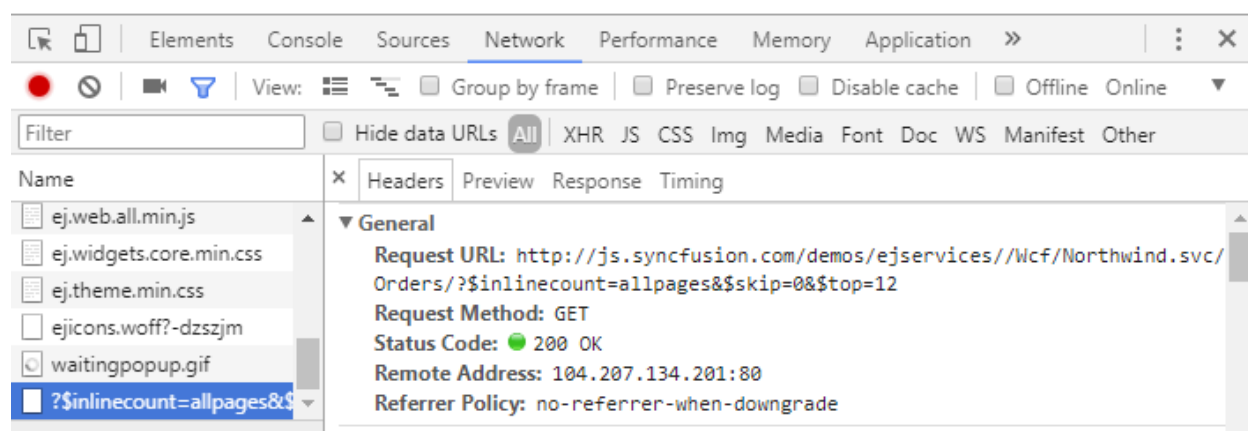
### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var dataManager = new
        ej.DataManager("http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/
        Orders/");
        var gridInstance = new ej.Grid($("#Grid"), {
            dataSource: dataManager,
            allowPaging:true,
            columns: ["OrderID", "EmployeeID", "CustomerID", "ShipCountry", "Freight"]
        });
    });
}
```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID | CustomerID | ShipCountry | Freight  |
|---------|------------|------------|-------------|----------|
| 10248   | 5          | VINET      | France      | 32.3800  |
| 10249   | 6          | TOMSP      | Germany     | 11.6100  |
| 10250   | 4          | HANAR      | Brazil      | 65.8300  |
| 10251   | 3          | VICTE      | France      | 41.3400  |
| 10252   | 4          | SUPRD      | Belgium     | 51.3000  |
| 10253   | 3          | HANAR      | Brazil      | 58.1700  |
| 10254   | 5          | CHOPS      | Switzerland | 22.9800  |
| 10255   | 9          | RICSU      | Switzerland | 148.3300 |
| 10256   | 3          | WELLI      | Brazil      | 13.9700  |
| 10257   | 4          | HILAA      | Venezuela   | 81.9100  |
| 10258   | 1          | ERNSH      | Austria     | 140.5100 |
| 10259   | 4          | CENTC      | Mexico      | 3.2500   |

1 2 3 4 5 6 7 8 ... 1 of 70 pages (830 items)



**Note:** By default, if no adaptor is specified for [ej.DataManager](#) and only the url link is mentioned it will consider as ODataService.

## Columns

Column definitions are used as the [dataSource](#) schema in Grid and it plays vital role in rendering column values in required format. Grid operations such as sorting, filtering, editing would be performed based

on the column definitions. The `field` property of the `columns` is necessary to map the datasource values in Grid columns.

**Note:** 1. If the column with [field](#) is not in the datasource, then the column values will be displayed as empty.

2. If the field name contains "dot" operator then it is considered as complex binding.

## Column Template

HTML templates can be specified in the [template](#) property of the particular column as a string (HTML element) or ID of the template's HTML element.

You can use JsRender syntax in the template. For more information about JsRender syntax, please refer [this link](#).

**Note:** If [field](#) is not specified, you will not able to perform editing, grouping, filtering, sorting, search and summary functionalities in particular column.

The following code example describes the above behavior.

## HTML

<div id="Grid"></div>

TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.employeeView" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["employeeView"],
allowPaging: true,
pageSettings: {
pageSize: 4
},
columns: [
{ headerText: "Photo", template: "<img style='width: 75px; height: 70px;'
src='Employees/{0}{1}{2}:EmployeeID{3}.png' />" },
{ field: "EmployeeID" },
{ field: "FirstName" },
{ field: "LastName" },
{ field: "Country" }
]
});
});
}
}

```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID | Freight | ShipCountry | ShipCity       |
|---------|------------|---------|-------------|----------------|
| 10248   | 5          | 32.38   | France      | Reims          |
| 10249   | 6          | 11.61   | Germany     | Münster        |
| 10250   | 4          | 65.83   | Brazil      | Rio de Janeiro |
| 10251   | 3          | 41.34   | France      | Lyon           |
| 10252   | 4          | 51.3    | Belgium     | Charleroi      |
| 10253   | 3          | 58.17   | Brazil      | Rio de Janeiro |
| 10254   | 5          | 22.98   | Switzerland | Bern           |
| 10255   | 9          | 148.33  | Switzerland | Genève         |
| 10256   | 3          | 13.97   | Brazil      | Resende        |
| 10257   | 4          | 81.91   | Venezuela   | San Cristóbal  |
| 10258   | 1          | 140.51  | Austria     | Graz           |
| 10259   | 4          | 3.25    | Mexico      | México D.F.    |

⏪

⏩

1

2

3

4

5

6

7

8

...

⏪

⏩

1 of 17 pages (200 items)

## Command Column

### Default action buttons

Using [command](#) column, you can add CRUD action buttons as one of the Grid column, through [type](#) property of [commands](#). The type property supports the below default [UnboundType](#) buttons.

1. edit
2. save
3. delete
4. cancel

Through [buttonOptions](#) property of [commands](#), you can specify all the button options which are supported by Essential Studio JavaScript [Button](#) control.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
```



```
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource : window.gridData,
allowPaging : true,
editSettings : {
allowEditing : true,
allowAdding : true,
allowDeleting : true
},
columns : [
{ field: "OrderID", isPrimaryKey: true },
{ field: "EmployeeID" },
{ field: "Freight", editType: "numericedit"},
{ field: "ShipCountry" },
{
headerText : "Manage Records",
commands : [
{ type : "edit", buttonOptions : { text : "Edit" } },
{ type : "delete", buttonOptions : { text : "Delete" } },
{ type : "save", buttonOptions : { text : "Save" } },
{ type : "cancel", buttonOptions : { text : "Cancel" } }
],
width : 150
}
]
});
});
}
```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID                     | Freight                 | ShipCountry                         | Manage Records                                |
|---------|--------------------------------|-------------------------|-------------------------------------|---|
| 10248   | 5                              | 32.38                   | France                              | <button>Edit</button> <button>Delete</button> |
| 10249   | 6                              | 11.61                   | Germany                             | <button>Edit</button> <button>Delete</button> |
| 10250   | 4                              | 65.83                   | Brazil                              | <button>Edit</button> <button>Delete</button> |
| 10251   | 3                              | 41.34                   | France                              | <button>Edit</button> <button>Delete</button> |
| 10252   | 4                              | 51.3                    | Belgium                             | <button>Edit</button> <button>Delete</button> |
| 10253   | <input type="text" value="3"/> | 58 <input type="text"/> | <input type="text" value="Brazil"/> | <button>Save</button> <button>Cancel</button> |
| 10254   | 5                              | 22.98                   | Switzerland                         | <button>Edit</button> <button>Delete</button> |
| 10255   | 9                              | 148.33                  | Switzerland                         | <button>Edit</button> <button>Delete</button> |
| 10256   | 3                              | 13.97                   | Brazil                              | <button>Edit</button> <button>Delete</button> |
| 10257   | 4                              | 81.91                   | Venezuela                           | <button>Edit</button> <button>Delete</button> |
| 10258   | 1                              | 140.51                  | Austria                             | <button>Edit</button> <button>Delete</button> |
| 10259   | 4                              | 3.25                    | Mexico                              | <button>Edit</button> <button>Delete</button> |

« ◀ 1 2 3 4 5 6 7 8 ... ▶ »
1 of 17 pages (200 items)

### Custom buttons

You can add custom button in the command column by specifying the [type](#) property of [commands](#) as "empty" or any other [string](#) which does not corresponds to default [UnboundType](#) buttons.

**Note:** 1. For [type](#) property you can assign either [string](#) value ("edit") or [enum](#) value (ej.Grid.UnboundType.Edit).

2. In command column you can add only buttons.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["employeeView"],
columns: [
{ field: "EmployeeID" },
{
headerText: "Employee Details",
commands: [
{ type: "details", buttonOptions: { text: "Details", width: "100", click:
"onClick" } }
],
textAlign: ej.TextAlign.Center,
width: 150
}
]
});
});
}
function onClick(args) {
let grid: ej.Grid = $("#Grid").ejGrid("instance");
let index = this.element.closest("tr").index();
let record = grid.getCurrentViewData()[index];
alert("Record Details: " + JSON.stringify(record));
}
```

The following output is displayed as a result of the above code example.

| EmployeeID | Employee Details         |
|------------|--------------------------|
| 1          | <button>Details</button> |
| 2          | <button>Details</button> |
| 3          | <button>Details</button> |
| 4          | <button>Details</button> |
| 5          | <button>Details</button> |
| 6          | <button>Details</button> |
| 7          | <button>Details</button> |
| 8          | <button>Details</button> |
| 9          | <button>Details</button> |

This page says:

Record Details: {"EmployeeID": 1, "LastName": "Davolio", "FirstName": "Nancy", "Title": "Sales Representative", "TitleOfCourtesy": "Ms.", "BirthDate": "1948-12-08T08:00:00.000Z", "HireDate": "1992-05-01T07:00:00.000Z", "Address": "507 - 20th Ave. E.\r\nApt. 2A", "City": "Seattle", "Region": "WA", "PostalCode": "98122", "Country": "USA", "HomePhone": "(206) 555-9857", "Extension": "5467", "Photo": {"Length": 21626}, "Notes": "Education includes a BA in psychology from Colorado State University in 1970. She also completed 'The Art of the Cold Call.' Nancy is a member of Toastmasters International.", "ReportsTo": 2, "PhotoPath": "http://accweb/emmployees/davolio.bmp"}

OK

### Column Chooser

Column chooser contains the list of all the columns which are defined in the [columns](#) property. Using this you can control the visibility of columns in Grid. You can prevent to show the particular column name in column chooser by setting [showInColumnChooser](#) property of [columns](#) as `false`.

Column Chooser would be shown in the top right corner of Grid. To enable column chooser, set [showColumnChooser](#) property as `true`.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.gridData" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            allowPaging: true,
            showColumnChooser: true,
            columns: [
                { field: "OrderID" },
            ]
        });
    });
}

```

```

{ field: "EmployeeID", showInColumnChooser: false },
{ field: "Freight" },
{ field: "ShipCity" },
{ field: "ShipCountry" }
]
});
});
}

```

The following output is displayed as a result of the above code example.

| Columns ▾ |            |         |                |             |
|-----------|------------|---------|----------------|-------------|
| OrderID   | EmployeeID | Freight | ShipCity       | ShipCountry |
| 10248     | 5          | 32.38   | Reims          |             |
| 10249     | 6          | 11.61   | Münster        |             |
| 10250     | 4          | 65.83   | Rio de Janeiro |             |
| 10251     | 3          | 41.34   | Lyon           |             |
| 10252     | 4          | 51.3    | Charleroi      |             |
| 10253     | 3          | 58.17   | Rio de Janeiro |             |
| 10254     | 5          | 22.98   | Bern           |             |
| 10255     | 9          | 148.33  | Genève         |             |
| 10256     | 3          | 13.97   | Resende        | Brazil      |
| 10257     | 4          | 81.91   | San Cristóbal  | Venezuela   |
| 10258     | 1          | 140.51  | Graz           | Austria     |
| 10259     | 4          | 3.25    | México D.F.    | Mexico      |

☒ (Select All)
 ☒ OrderID
 ☒ Freight
 ☒ ShipCity
 ☒ ShipCountry
 Done Cancel

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

### Foreign Key Column

Lookup data source can be bound to [dataSource](#) property of [columns](#). Data [field](#) and [text](#) can be set using [foreignKeyField](#) and [foreignKeyValue](#) property of [columns](#).

In the [dataSource](#) property, we can bound local and remote data.

**Information:** For foreign key column the sorting and grouping is based on [foreignKeyField](#) instead of [foreignKeyValue](#).

**Note:** In remote data, server should be configured to perform select and filter operations since the Grid will try to fetch required columns using select operation and required data using filter operation.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

## TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" and "window.employeeView" is referred
from 'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
allowPaging: true,
editSettings: {
allowEditing: true,
allowAdding: true,
allowDeleting: true
},
columns: [
{ field: "OrderID", isPrimaryKey: true },
{ field: "EmployeeID", foreignKeyField: "EmployeeID", foreignKeyValue:
"FirstName", dataSource: window["employeeView"], headerText: "First Name" },
//(or)
{ field: "EmployeeID", foreignKeyField: "EmployeeID", foreignKeyValue:
"FirstName", dataSource: new ej.DataManager({ url:
"http://mvc.syncfusion.com/Services/Northwnd.svc/Employees/" }), headerText:
"First Name" },
{ field: "CustomerID" },
{ field: "Freight" },
{ field: "ShipCity" }
]
});
});
}
```

The following output is displayed as a result of the above code example.

| OrderID | First Name | CustomerID | Freight | ShipCity       |
|---------|------------|------------|---------|----------------|
| 10248   | Steven     | VINET      | 32.38   | Reims          |
| 10249   | Michael    | TOMSP      | 11.61   | Münster        |
| 10250   | Margaret   | HANAR      | 65.83   | Rio de Janeiro |
| 10251   | Janet      | VICTE      | 41.34   | Lyon           |
| 10252   | Margaret   | SUPRD      | 51.3    | Charleroi      |
| 10253   | Nancy      | HANAR      | 58.17   | Rio de Janeiro |
| 10254   | Andrew     | CHOPS      | 22.98   | Bern           |
| 10255   | Janet      | RICSU      | 148.33  | Genève         |
| 10256   | Margaret   | WELLI      | 13.97   | Resende        |
| 10257   | Steven     | HILAA      | 81.91   | San Cristóbal  |
| 10258   | Nancy      | ERNSH      | 140.51  | Graz           |
| 10259   | Margaret   | CENTC      | 3.25    | México D.F.    |

1

2

3

4

5

6

7

8

...

1 of 17 pages (200 items)

## Row

It represents the record details that are fetched from the datasource.

## Details Template

It provides a detailed view /additional information about each row of the grid. You can render any type of JsRender template and assign the script template id in the [detailsTemplate](#) property. And also you can change the HTML elements in detail template row into JavaScript controls using the [detailsDataBound](#) event.

On enabling details template, new column will be added in grid with an expander button in it and that can be expanded or collapsed to show or hide the underlying details of row respectively.

**Note:** It's a standard way to enclose the template within the `script` tag with `type` as "text/x-jsrender".

The following code example describes the above behavior.

## HTML

```
<div id="Grid"></div>
<script id="tabGridContents" type="text/x-jsrender">
<div class="tabcontrol" id="Test">
<ul>
<li><a href="#gridTab{{{"{":EmployeeID{}}}}">Stock Grid</a></li>
</ul>
<div id="gridTab{{:EmployeeID}}">
<div id="detailGrid">
</div>
</div>
```

```
</div>
</script>
```

## TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.employeeView" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["employeeView"],
            detailsTemplate: "#tabGridContents",
            detailsDataBound: detailGridData,
            columns: ["EmployeeID", "FirstName", "Title", "City", "Country"]
        });
    });
}

function detailGridData(e: ej.Grid.DetailsDataBoundEventArgs) {
    // Here you can get the parent details from "data". EmployeeID is the unique
    column value in parent row.
    var filteredData = e.rowData["EmployeeID"];
    // the datasource "window.ordersView" is referred from
    'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
    var data = new ej.DataManager(window["ordersView"]).executeLocal(new
    ej.Query().where("EmployeeID", ej.FilterOperators.equal,
    parseInt(filteredData), true).take(5)); // form the query to filter the
    detail row data by using EmployeeID column value.
    //detailsElement contains all the elements which are mentioned in the
    template.
    // Here the detailGrid element is changed as ejGrid control
    new ej.Grid($("#detailGrid"), {
        dataSource: data,
        columns: ["OrderID", "EmployeeID", "ShipCity", "ShipCountry", "Freight"]
    });
    // Here the element which has tabcontrol class is changed as ejTab control
    new ej.Tab($(".tabcontrol"));
}
```

The following output is displayed as a result of the above code example.



|                       | EmployeeID | FirstName              | Title       | City        | Country |
|-----------------------|------------|------------------------|-------------|-------------|---------|
| 1                     | Nancy      | Sales Representative   | Seattle     | USA         |         |
| <div>Stock Grid</div> |            |                        |             |             |         |
|                       | OrderID    | EmployeeID             | ShipCity    | ShipCountry | Freight |
|                       | 10835      | 1                      | Berlin      | Germany     | 69.53   |
|                       | 10952      | 1                      | Berlin      | Germany     | 40.42   |
|                       | 10677      | 1                      | México D.F. | Mexico      | 4.03    |
|                       | 10558      | 1                      | Colchester  | UK          | 72.97   |
|                       | 10453      | 1                      | Colchester  | UK          | 25.36   |
| 2                     | Andrew     | Vice President, Sales  | Tacoma      | USA         |         |
| 3                     | Janet      | Sales Representative   | Kirkland    | USA         |         |
| 4                     | Margaret   | Sales Representative   | Redmond     | USA         |         |
| 5                     | Steven     | Sales Manager          | London      | UK          |         |
| 6                     | Michael    | Sales Representative   | London      | UK          |         |
| 7                     | Robert     | Sales Representative   | London      | UK          |         |
| 8                     | Laura      | Inside Sales Coordinat | Seattle     | USA         |         |
| 9                     | Anne       | Sales Representative   | London      | UK          |         |

### Row Template

Row template enables you to set the customized look and behavior to all Grid rows. The [rowTemplate](#) property can be used to bind the `id` of HTML template.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
<script id="templateData" type="text/x-jsrender">
<tr>
<td class="photo">
 />
</td>
<td class="details">
<table class="CardTable" cellpadding="3" cellspacing="2">
<colgroup>
<col width="50%">
```

```

<col width="50%">
</colgroup>
<tbody>
<tr>
<td class="CardHeader">First Name </td>
<td>{{:FirstName}}</td>
</tr>
<tr>
<td class="CardHeader">Last Name</td>
<td>{{:LastName}}</td>
</tr>
<tr>
<td class="CardHeader">Title
</td>
<td>{{:Title}}</td>
</tr>
</tbody>
</table>
</td>
</tr>
</script>

```

## CSS

```

.photo img {
width: 130px;
}
.photo, .details {
border-color: #c4c4c4;
border-style: solid;
}
.photo {
border-width: 1px 0px 0px 0px;
}
.details {
border-width: 1px 0px 0px 1px;
}
.details > table {
width: 100%;
}
.CardHeader {
font-weight: bolder;
}

```

## TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.employeeView" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: new ej.DataManager(window["employeeView"]).executeLocal(new
ej.Query().take(2)),
rowTemplate: "#templateData", // row template

```

```

columns: [
{ headerText: "Photo", width: 30 },
{ headerText: 'Employee Details', width: 70 }
]
});
});
}

```

The following output is displayed as a result of the above code example.

| Photo  | Employee Details  |
|--|---|
|   | <div>First Name</div> <div>Last Name</div> <div>Title</div> <div>Nancy Davolio</div> <div>Sales Representative</div>  |
|  | <div>First Name</div> <div>Last Name</div> <div>Title</div> <div>Andrew Fuller</div> <div>Vice President, Sales</div> |

### Drag-and-Drop

The Grid rows can be reordered, dropped to another Grid or custom control by enabling the [allowRowDragAndDrop](#) Grid property.

**Note:** To enable selection of multiple rows by mouse dragging on Grid rows, the [selectionType](#) property of Grid must be set to **multiple**.

### Reorder

By simply enabling the property [allowRowDragAndDrop](#), Grid rows can be reordered within the same Grid.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {

```

```
// the datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
allowPaging: true,
allowRowDragAndDrop: true,
selectionType: "multiple",
columns: [
{ field: "OrderID", headerText: "Order ID", isPrimaryKey: true, textAlign:
ej.TextAlign.Right, width: 80 },
{ field: "CustomerID", headerText: "Customer ID", width: 90 },
{ field: "Freight", headerText: "Freight", textAlign: ej.TextAlign.Right,
width: 75, format: "{0:C}" },
{ field: "ShipCountry", headerText: "Ship Country", width: 110 }
]
});
});
}
```

The following output is displayed before reordering rows.

| Order ID | Customer ID | Freight  | Ship Country |
|----------|-------------|----------|--------------|
| 10248    | VINET       | \$32.38  | France       |
| 10249    | TOMSP       | \$11.61  | Germany      |
| 10250    | HANAR       | \$65.83  | Brazil       |
| 10251    | VICTE       | \$41.34  | France       |
| 10252    | SUPRD       | \$51.30  | Belgium      |
| 10253    | HANAR       | \$58.17  | Brazil       |
| 10254    | CHOPS       | \$22.98  | Switzerland  |
| 10255    | RICSU       | \$148.33 | Switzerland  |
| 10256    | WELLI       | \$13.97  | Brazil       |
| 10257    | HILAA       | \$81.91  | Venezuela    |
| 10258    | ERNSH       | \$140.51 | Austria      |
| 10259    | CENTC       | \$3.25   | Mexico       |

1 of 17 pages (200 items)

The following output is displayed after reordering rows.

| Order ID | Customer ID | Freight  | Ship Country |
|----------|-------------|----------|--------------|
| 10248    | VINET       | \$32.38  | France       |
| 10252    | SUPRD       | \$51.30  | Belgium      |
| 10253    | HANAR       | \$58.17  | Brazil       |
| 10254    | CHOPS       | \$22.98  | Switzerland  |
| 10249    | TOMSP       | \$11.61  | Germany      |
| 10250    | HANAR       | \$65.83  | Brazil       |
| 10251    | VICTE       | \$41.34  | France       |
| 10255    | RICSU       | \$148.33 | Switzerland  |
| 10256    | WELLI       | \$13.97  | Brazil       |
| 10257    | HILAA       | \$81.91  | Venezuela    |
| 10258    | ERNSH       | \$140.51 | Austria      |
| 10259    | CENTC       | \$3.25   | Mexico       |

1
2
3
4
5
6
7
8
...

1 of 17 pages (200 items)

### Grid-to-Grid

To drag and drop rows between two Grid, enable the Grid property [allowRowDragAndDrop](#) and specify the target Grid ID in [dropTargetID](#) property of the Grid [rowDropSettings](#).

The following code example describes the above behavior.

### HTML

```
<div id="Grid" style="float:left;width:49%"></div>
<div id="DestinationGrid" style="float:right;width:49%"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var dataManager = new
ej.DataManager("http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/
Orders/");
var gridInstance = new ej.Grid($("#Grid"), {
```

```
// the datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
allowPaging: true,
allowRowDragAndDrop: true,
selectionType: "multiple",
rowDropSettings: { dropTargetID: "#DestinationGrid" },
columns: [
{ field: "OrderID", headerText: "Order ID", isPrimaryKey: true, textAlign:
ej.TextAlign.Right, width: 80 },
{ field: "CustomerID", headerText: "Customer ID", width: 90 },
{ field: "Freight", headerText: "Freight", textAlign: ej.TextAlign.Right,
width: 75, format: "{0:C}" },
{ field: "ShipCountry", headerText: "Ship Country", width: 110 }
]
});
var destGridInstance = new ej.Grid($("#DestinationGrid"), {
dataSource: [],
allowPaging: true,
allowRowDragAndDrop: true,
rowDropSettings: { dropTargetID: "#Grid" },
selectionType: "multiple",
columns: [
{ field: "OrderID", headerText: "Order ID", isPrimaryKey: true, textAlign:
ej.TextAlign.Right, width: 80 },
{ field: "CustomerID", headerText: "Customer ID", width: 90 },
{ field: "Freight", headerText: "Freight", textAlign: ej.TextAlign.Right,
width: 75, format: "{0:C}" },
{ field: "ShipCountry", headerText: "Ship Country", width: 110 }
]
});
});
});
}
```

The following output is displayed before dropping Grid rows.

| Order ID | Customer ID | Freight  | Ship Country |
|----------|-------------|----------|--------------|
| 10248    | VINET       | \$32.38  | France       |
| 10249    | TOMSP       | \$11.61  | Germany      |
| 10250    | HANAR       | \$65.83  | Brazil       |
| 10251    | VICTE       | \$41.34  | France       |
| 10252    | SUPRD       | \$51.30  | Belgium      |
| 10253    | HANAR       | \$58.17  | Brazil       |
| 10254    | CHOPS       | \$22.98  | Switzerland  |
| 10255    | RICSU       | \$148.33 | Switzerland  |
| 10256    | WELLI       | \$13.97  | Brazil       |
| 10257    | HILAA       | \$81.91  | Venezuela    |
| 10258    | ERNSH       | \$140.51 | Austria      |
| 10259    | CENTC       | \$3.25   | Mexico       |

| Order ID              | Customer ID | Freight | Ship Country |
|-----------------------|-------------|---------|--------------|
| No records to display |             |         |              |
| 10249                 | TOMSP       | \$11.61 | Germany      |
| 10250                 | HANAR       | \$65.83 | Brazil       |

The following output is displayed after dropping Grid rows.

| Order ID | Customer ID | Freight  | Ship Country |
|----------|-------------|----------|--------------|
| 10248    | VINET       | \$32.38  | France       |
| 10251    | VICTE       | \$41.34  | France       |
| 10252    | SUPRD       | \$51.30  | Belgium      |
| 10253    | HANAR       | \$58.17  | Brazil       |
| 10254    | CHOPS       | \$22.98  | Switzerland  |
| 10255    | RICSU       | \$148.33 | Switzerland  |
| 10256    | WELLI       | \$13.97  | Brazil       |
| 10257    | HILAA       | \$81.91  | Venezuela    |
| 10258    | ERNSH       | \$140.51 | Austria      |
| 10259    | CENTC       | \$3.25   | Mexico       |
| 10260    | OTTIK       | \$55.09  | Germany      |
| 10261    | QUEDE       | \$3.05   | Brazil       |

1 of 17 pages (198 items)

| Order ID | Customer ID | Freight | Ship Country |
|----------|-------------|---------|--------------|
| 10249    | TOMSP       | \$11.61 | Germany      |
| 10250    | HANAR       | \$65.83 | Brazil       |

1 of 1 pages (2 items)

### Grid-to-Custom control

You can also drag and drop Grid rows to any custom control. For instance, let it be a form.

Enable the Grid property [allowRowDragAndDrop](#) and specify the target form element ID in [dropTargetID](#) property of Grid [rowDropSettings](#).

The following code example describes the above behavior.

### HTML

```
<div id="Grid" style="float:left;width:60%"></div>
<div style="float:right;width:38%">
<form role="form" id="dropForm" style="width:98%">
<fieldset style="text-align:center; font-weight:700"><legend>Record
Details</legend></fieldset>
<div class="form-group row">
<label for="OrderID">Order ID:</label>
<input class="form-control" name="OrderID">
</div>
<div class="form-group row">
<label for="CustomerID">Customer ID:</label>
<input name="CustomerID" class="form-control">
</div>
<div class="form-group row">
<label for="EmployeeID">Employee ID:</label>
<input name="EmployeeID" class="form-control">
</div>
<div class="form-group row">
<label for="Freight">Freight:</label>
<input name="Freight" class="form-control">
</div>
<div class="form-group row">
<label for="ShipCity">Ship City:</label>
<input name="ShipCity" class="form-control">
</div>
<br />
</form>
</div>
```

**TS**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
// the datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: new ej.DataManager(window["gridData"]),
allowPaging: true,
allowRowDragAndDrop: true,
rowDropSettings: { dropTargetID: "#dropForm" },
rowDrop: rowDropHandler,
columns: [
{ field: "OrderID", headerText: "Order ID", isPrimaryKey: true, textAlign:
ej.TextAlign.Right, width: 80 },
{ field: "CustomerID", headerText: "Customer ID", width: 120 },
{ field: "EmployeeID", headerText: "Employee ID", textAlign:
ej.TextAlign.Right, width: 120 },
{ field: "Freight", headerText: "Freight", textAlign: ej.TextAlign.Right,
width: 75, format: "{0:C}" },
{ field: "ShipCity", headerText: "Ship City", width: 140 }
]
});
});
}
function rowDropHandler(e: ej.Grid.RowDropEventArgs) {
for (var key in e.data[0]) {
$('#dropForm input[name=' + key + ']').val(e.data[0][key]);
}
}
}
```

The following output is displayed before dropping the rows on Form.



| Order ID | Customer ID | Employee ID | Freight  | Ship City      |
|----------|-------------|-------------|----------|----------------|
| 10248    | VINET       | 5           | \$32.38  | Reims          |
| 10249    | TOMSP       | 6           | \$11.61  | Münster        |
| 10250    | HANAR       | 4           | \$65.83  | Rio de Janeiro |
| 10251    | VICTE       | 3           | \$41.34  | Lyon           |
| 10252    | SUPRD       | 4           | \$51.30  | Charleroi      |
| 10253    | HANAR       | 3           | \$58.17  | Rio de Janeiro |
| 10254    | CHOPS       | 5           | \$22.98  | Bern           |
| 10255    | RICSU       | 9           | \$148.33 | Genève         |
| 10256    | WELLI       | 3           | \$13.97  | Resende        |
| 10257    | HILAA       | 4           | \$81.91  | San Cristóbal  |
| 10258    | ERNSH       | 1           | \$140.51 | Graz           |
| 10259    | CENTC       | 4           | \$3.25   | México D.F.    |

1 of 17 pages (200 items)

### Record Details

Order ID:

Customer ID:

Employee ID:

Freight:

Ship City:

The following output is displayed after dropping the rows on Form.

| Order ID | Customer ID | Employee ID | Freight  | Ship City      |
|----------|-------------|-------------|----------|----------------|
| 10248    | VINET       | 5           | \$32.38  | Reims          |
| 10249    | TOMSP       | 6           | \$11.61  | Münster        |
| 10250    | HANAR       | 4           | \$65.83  | Rio de Janeiro |
| 10251    | VICTE       | 3           | \$41.34  | Lyon           |
| 10252    | SUPRD       | 4           | \$51.30  | Charleroi      |
| 10253    | HANAR       | 3           | \$58.17  | Rio de Janeiro |
| 10254    | CHOPS       | 5           | \$22.98  | Bern           |
| 10255    | RICSU       | 9           | \$148.33 | Genève         |
| 10256    | WELLI       | 3           | \$13.97  | Resende        |
| 10257    | HILAA       | 4           | \$81.91  | San Cristóbal  |
| 10258    | ERNSH       | 1           | \$140.51 | Graz           |
| 10259    | CENTC       | 4           | \$3.25   | México D.F.    |

1 of 17 pages (200 items)

### Record Details

Order ID:

Customer ID:

Employee ID:

Freight:

Ship City:

**Note:** The default behavior of drag and drop between Grid or any other controls is as cut and paste. For copy and paste behavior specify the drag behavior in [dragBehavior](#) property of the [rowDropSettings](#) as `ej.Grid.DragBehavior.Copy`.

## Editing

The grid control has support for the dynamic insertion, updating and deletion of records. You can start the edit action either by double clicking the particular row or by selecting the required row and clicking on Edit icon in toolbar. Similarly, you can add new record to grid either by clicking on insert icon in

toolbar or on an external button which is bound to call [addRecord](#) method of grid. **Save** and **Cancel** while on edit mode is possible using respective toolbar icon in grid.

Deletion of the record is possible by selecting the required row and clicking on Delete icon in toolbar.

The primary key for the data source should be defined in [columns](#) definition, for editing to work properly. In [columns](#) definition, particular primary column's [isPrimaryKey](#) property should be set to **true**. Refer to the Knowledge base [link](#) for more information.

**Note:** 1. In grid, the primary key column will be automatically set to read only while editing the row, but you can specify primary key column value while adding a new record.

**Note:** 2. The column which is specified as [isIdentity](#) will be in readonly mode both while editing and adding a record. Also, auto incremented value is assigned to that [isIdentity](#) column.

#### Toolbar with edit option

Using toolbar which is rendered at the top of the grid header, you can show all the CRUD related action. To enable toolbar and toolbar items, set [showToolbar](#) property as true and [toolbarItems](#). The default toolbar items are **Add**, **Edit**, **Delete**, **Update** and **Cancel**.

**Note:** For [toolbarItems](#) property you can assign either **string** value ("add") or **enum** value (`ej.Grid.ToolBarItems.Add`).

The following code example describes the above behavior.

#### HTML

```
<div id="Grid"></div>
```

#### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.gridData" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            toolbarSettings: { showToolbar: true, toolbarItems: ["add", "edit",
                "delete", "update", "cancel"] },
            editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true
            },
            allowPaging: true,
            columns: [
                { field: "OrderID", isPrimaryKey: true },
                { field: "CustomerID" },
                { field: "EmployeeID" },
                { field: "ShipCity" },
                { field: "ShipCountry" }
            ]
        });
    });
}
```

The following output is displayed as a result of the above code example.

| OrderID | CustomerID | EmployeeID | ShipCity       | ShipCountry |
|---------|------------|------------|----------------|-------------|
| 10248   | VINET      | 5          | Reims          | France      |
| 10249   | TOMSP      | 6          | Münster        | Germany     |
| 10250   | HANAR      | 4          | Rio de Janeiro | Brazil      |
| 10251   | VICTE      | 3          | Lyon           | France      |
| 10252   | SUPRD      | 4          | Charleroi      | Belgium     |
| 10253   | HANAR      | 3          | Rio de Janeiro | Brazil      |
| 10254   | CHOPS      | 5          | Bern           | Switzerland |
| 10255   | RICSU      | 9          | Genève         | Switzerland |
| 10256   | WELLI      | 3          | Resende        | Brazil      |
| 10257   | HILAA      | 4          | San Cristóbal  | Venezuela   |
| 10258   | ERNSH      | 1          | Graz           | Austria     |
| 10259   | CENTC      | 4          | México D.F.    | Mexico      |

1 of 17 pages (200 items)

### Cell edit type and its params

The edit type of bound column can be customized using [editType](#) property of [columns](#). The following Essential JavaScript controls are supported built-in by [editType](#). You can set the [editType](#) based on specific data type of the column.

- [CheckBox](#) control for boolean data type.
- [NumericTextBox](#) control for integers, double, and decimal data types.
- [InputTextBox](#) control for string data type.
- [DatePicker](#) control for date data type.
- [DateTimePicker](#) control for date-time data type.
- [DropDownList](#) control for list of data type.

And also you can define the model for all the editTypes controls while editing through [editParams](#) property of [columns](#).

The following table describes [editType](#) and their corresponding [editParams](#) of the specific data type of the column.

| EditType | EditParams                 | Example                       |
|----------|----------------------------|-------------------------------|
| CheckBox | <a href="#">ejCheckBox</a> | editParams: { checked: true } |

|                |                                  |   |
|----------------|----------------------------------|---|
| NumericTextBox | <a href="#">ejTextBoxes</a>      | editParams: { decimalPlaces: 2, value:5 } |
| InputTextBox   | -                                | -   |
| DatePicker     | <a href="#">ejDatePicker</a>     | editParams: { buttonText : "Now" }        |
| DateTimePicker | <a href="#">ejDateTimePicker</a> | editParams: { enabled: true }             |
| DropDownList   | <a href="#">ejDropDownList</a>   | editParams: { allowGrouping: true }       |

**Note:** 1. If [editType](#) is not set, then by default it will display the input element ("stringedit") while editing a column.

**Note:** 2. For [editType](#) property you can assign either `string` value ("numericedit") or `enum` value (ej.Grid.EditingType.Numeric).

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.gridData" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            toolbarSettings: { showToolbar: true, toolbarItems: ["add", "edit",
                "delete", "update", "cancel"] },
            editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true
            },
            allowPaging: true,
            columns: [
                { field: "OrderID", isPrimaryKey: true },
                { field: "CustomerID", editType: "stringedit" },
                { field: "Freight", editType: "numericedit", editParams: { decimalPlaces: 2
                } },
                { field: "ShipCity", editType: "dropdownedit", editParams: {
                enableAnimation: true } },
                { field: "ShipCountry" },
                { field: "OrderDate", editType: "datepicker", format: "{0:MM/dd/yyyy}",
                editParams: { buttonText: "Now" } },
                { field: "Verified", editType: "booleanedit", editParams: {
                showRoundedCorner: true } }
            ]
        });
    });
}
```

The following output is displayed as a result of the above code example.

| OrderID | CustomerID | Freight | ShipCity       | ShipCountry | OrderDate   | Verified                            |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
|---------|------------|---------|----------------|-------------|---|-------------------------------------|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|
| 10248   | VINET      | 32.38   | Reims          | France      | 07/04/1996  | <input checked="" type="checkbox"/> |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 10249   | TOMSP      | 11.61   | Münster        | Germany     | 07/05/1996  | <input type="checkbox"/>            |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 10250   | HANAR      | 65.83   | Rio de Janeiro | Brazil      | 07/08/1996  | <input checked="" type="checkbox"/> |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 10251   | VICTE      | 41.34   | Lyon           | France      | 07/08/1996  | <input checked="" type="checkbox"/> |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 10252   | SUPRD      | 51.3    | Charleroi      | Belgium     | <div> <div>July 1996</div> <table> <tr> <th>Su</th><th>Mo</th><th>Tu</th><th>We</th><th>Th</th><th>Fr</th><th>Sa</th></tr> <tr> <td>30</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr> <td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td></tr> <tr> <td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td></tr> <tr> <td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td></tr> <tr> <td>28</td><td>29</td><td>30</td><td>31</td><td>1</td><td>2</td><td>3</td></tr> </table> <div>Now</div> </div> |                                     | Su | Mo | Tu | We | Th | Fr | Sa | 30 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 1 | 2 | 3 |
| Su      | Mo         | Tu      | We             | Th          | Fr  | Sa                                  |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 30      | 1          | 2       | 3              | 4           | 5   | 6                                   |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 7       | 8          | 9       | 10             | 11          | 12  | 13                                  |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 14      | 15         | 16      | 17             | 18          | 19  | 20                                  |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 21      | 22         | 23      | 24             | 25          | 26  | 27                                  |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 28      | 29         | 30      | 31             | 1           | 2   | 3                                   |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 10253   | HANAR      | 58.17   | Rio de Janeiro | Brazil      |   |                                     |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 10254   | CHOPS      | 22.98   | Bern           | Switzerland |   |                                     |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 10255   | RICSU      | 148.33  | Genève         | Switzerland |   |                                     |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 10256   | WELLI      | 13.97   | Resende        | Brazil      |   |                                     |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 10257   | HILAA      | 81.91   | San Cristóbal  | Venezuela   |   |                                     |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 10258   | ERNSH      | 140.51  | Graz           | Austria     |   |                                     |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |
| 10259   | CENTC      | 3.25    | México D.F.    | Mexico      | 07/18/1996  | <input type="checkbox"/>            |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |

1

2

3

4

5

6

7

8

...

1 of 17 pages (200 items)

### Cell Edit Template

On editing the column values, custom editor can be created by using the [editTemplate](#) property of [columns](#). It has three functions, they are

1. **create** - It is used to create the control at time of initializing.
2. **read** - It is used to read the input value at time of saving.
3. **write** - It is used to assign the value to control at time of editing.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
  $(function () {
    var gridInstance = new ej.Grid($("#Grid"), {
      //The datasource "window.gridData" is referred from
      'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
```

```
dataSource: window["gridData"],
toolbarSettings: { showToolbar: true, toolbarItems: ["add", "edit",
"delete", "update", "cancel"] },
editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true
},
allowPaging: true,
columns: [
{ field: "OrderID", isPrimaryKey: true },
{ field: "CustomerID" },
{ field: "Freight" },
{ field: "ShipCountry" },
{
field: "ShipPostalCode",
editTemplate: {
create: function () {
return "<input>";
},
read: function (args) {
return args.ejMaskEdit("get_UnstrippedValue");
},
write: function (args) {
args.element.ejMaskEdit({
maskFormat: "99-99-9999",
value: args.rowdata["ShipPostalCode"]
});
}
}
}
]
});
});
}
```

The following output is displayed as a result of the above code example.

| OrderID | CustomerID | Freight | ShipCountry | ShipPostalCode |
|---------|------------|---------|-------------|----------------|
| 10248   | VINET      | 32.38   | France      | 51100          |
| 10249   | TOMSP      | 11.61   | Germany     | 44-08-7        |
| 10250   | HANAR      | 65.83   | Brazil      | 05454-876      |
| 10251   | VICTE      | 41.34   | France      | 69004          |
| 10252   | SUPRD      | 51.3    | Belgium     | B-6000         |
| 10253   | HANAR      | 58.17   | Brazil      | 05454-876      |
| 10254   | CHOPS      | 22.98   | Switzerland | 3012           |
| 10255   | RICSU      | 148.33  | Switzerland | 1204           |
| 10256   | WELLI      | 13.97   | Brazil      | 08737-363      |
| 10257   | HILAA      | 81.91   | Venezuela   | 5022           |
| 10258   | ERNSH      | 140.51  | Austria     | 8010           |
| 10259   | CENTC      | 3.25    | Mexico      | 05022          |

1 of 17 pages (200 items)

## Edit Modes

### Inline

Set [editMode](#) as **normal**, then the row itself is changed as edited row.

**Note:** For [editMode](#) property you can assign either **string** value ("normal") or **enum** value (`ej.Grid.EditMode.Normal`).

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
  $(function () {
    var gridInstance = new ej.Grid($("#Grid"), {
      //The datasource "window.gridData" is referred from
      'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
      dataSource: window["gridData"],
      toolbarSettings: {
        showToolbar: true,
```

```

toolbarItems: ["add", "edit", "delete", "update", "cancel"]
},
editSettings: {
allowEditing: true,
allowAdding: true,
allowDeleting: true,
editMode: "normal"
},
allowPaging: true,
columns: [
{ field: "OrderID", isPrimaryKey: true },
{ field: "CustomerID" },
{ field: "Freight", editType: "numericedit" },
{ field: "ShipCountry", editType: "dropdownedit" },
{ field: "OrderDate", editType: "datepicker", format: "{0:dd/MM/yyyy}" }
]
});
});
}

```

The following output is displayed as a result of the above code example.

| <div> <div>+</div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> |            |         |             |            |
|---|------------|---------|-------------|------------|
| OrderID   | CustomerID | Freight | ShipCountry | OrderDate  |
| 10248   | VINET      | 32.38   | France      | 04/07/1996 |
| 10249   | TOMSP      | 11.61   | Germany     | 05/07/1996 |
| 10250   | HANAR      | 65.83   | Brazil      | 08/07/1996 |
| 10251   | VICTE      | 41.34   | France      | 08/07/1996 |
| 10252   | SUPRD      | 51      | Belgium     | 09/07/1996 |
| 10253   | HANAR      | 58.17   | Argentina   | 10/07/1996 |
| 10254   | CHOPS      | 22.98   | Austria     | 11/07/1996 |
| 10255   | RICSU      | 148.33  | Belgium     | 12/07/1996 |
| 10256   | WELLI      | 13.97   | Brazil      | 15/07/1996 |
| 10257   | HILAA      | 81.91   | Canada      | 16/07/1996 |
| 10258   | ERNSH      | 140.51  | Austria     | 17/07/1996 |
| 10259   | CENTC      | 3.25    | Mexico      | 18/07/1996 |

⏪

⏴

1

2

3

4

5

6

7

8

...

⏵

⏩

1 of 17 pages (200 items)

#### Inline Form

Set `editMode` as `inlineform`, then edit form will be inserted next to the row which is to be edited.



The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.gridData" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            toolbarSettings: {
                showToolbar: true,
                toolbarItems: ["add", "edit", "delete", "update", "cancel"]
            },
            editSettings: {
                allowEditing: true,
                allowAdding: true,
                allowDeleting: true,
                editMode: "inlineform"
            },
            columns: [
                { field: "OrderID", isPrimaryKey: true },
                { field: "CustomerID" },
                { field: "Freight", editType: "numericedit" },
                { field: "ShipCountry", editType: "dropdownedit" },
                { field: "OrderDate", editType: "datepicker", format: "{0:dd/MM/yyyy}" }
            ]
        });
    });
}
```

The following output is displayed as a result of the above code example.

| OrderID | CustomerID | Freight | ShipCountry | OrderDate  |
|---------|------------|---------|-------------|------------|
| 10248   | VINET      | 32.38   | France      | 04/07/1996 |
| 10249   | TOMSP      | 11.61   | Germany     | 05/07/1996 |
| 10250   | HANAR      | 65.83   | Brazil      | 08/07/1996 |
| 10251   | VICTE      | 41.34   | France      | 08/07/1996 |

**Details of 10251**

OrderID

CustomerID

Freight

ShipCountry

OrderDate

|       |       |        |             |            |
|-------|-------|--------|-------------|------------|
| 10252 | SUPRD | 51.3   | Belgium     | 09/07/1996 |
| 10253 | HANAR | 58.17  | Brazil      | 10/07/1996 |
| 10254 | CHOPS | 22.98  | Switzerland | 11/07/1996 |
| 10255 | RICSU | 148.33 | Switzerland | 12/07/1996 |
| 10256 | WELLI | 13.97  | Brazil      | 15/07/1996 |
| 10257 | HILAA | 81.91  | Venezuela   | 16/07/1996 |

### Inline Template Form

You can edit any of the fields pertaining to a single record of data and apply it to a template so that the same format is applied to all the other records that you may edit later.

Using this template support, you can edit the fields that are not bound to grid columns.

To edit the records using Inline template form, set `editMode` as `inlineformtemplate` and specify the template ID to `editSettings.inlineFormTemplateID` property.

While using template form, you can change the HTML elements to appropriate JS controls based on the column type. This can be achieved by using `actionComplete` event of grid.

**Note:** 1. `value` attribute is used to bind the corresponding field value while editing.

**Note:** 2. `name` attribute is used to get the changed field values while saving the edited record.

**Note:** 3. It's a standard way to enclose the `template` within the `script` tag with `type` as "text/x-jsrender".

**Note:** 4. For `editMode` property you can assign either string value ("inlineformtemplate") or enum value (`ej.Grid.EditMode.InlineTemplateForm`)

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
<script id="template" type="text/template">
<table cellpadding="10">
<tr>
<td>Order ID</td>
<td>
<input id="OrderID" name="OrderID" disabled="disabled"
value="{{{"{":{}:OrderID{}}}}}" />
</td>
<td>Customer ID</td>
<td>
<input id="CustomerID" name="CustomerID" value="{{{"{":{}:CustomerID{}}}}}"
class="e-field e-ejinputtext" style="width: 116px; height: 28px" />
</td>
</tr>
<tr>
<td>Employee ID</td>
<td>
<input type="text" id="EmployeeID" name="EmployeeID"
value="{{{"{":{}:EmployeeID{}}}}}" />
</td>
<td>Ship City</td>
<td>
<select id="ShipCity" name="ShipCity">
<option value="Argentina">Argentina</option>
<option value="Austria">Austria</option>
<option value="Belgium">Belgium</option>
<option value="Brazil">Brazil</option>
<option value="Canada">Canada</option>
<option value="Denmark">Denmark</option>
</select>
</td>
</tr>
</table>
</script>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
toolbarSettings: {
showToolbar: true,
toolbarItems: ["add", "edit", "delete", "update", "cancel"]
},
},
```

```
editSettings: {
  allowEditing: true,
  allowAdding: true,
  allowDeleting: true,
  editMode: "inlineformtemplate",
  inlineFormTemplateID: "#template"
},
columns: [
  { field: "OrderID", isPrimaryKey: true },
  { field: "CustomerID" },
  { field: "ShipCity" }
],
actionComplete: complete
});
});
}
function complete(args: ej.Grid.ActionCompleteEventArgs) {
$("#EmployeeID").ejNumericTextbox();
$("#Freight").ejNumericTextbox();
$("#ShipCity").ejDropDownList();
}
```

The following output is displayed as a result of the above code example.

| OrderID | CustomerID | ShipCity       |
|---------|------------|----------------|
| 10248   | VINET      | Reims          |
| 10249   | TOMSP      | Münster        |
| 10250   | HANAR      | Rio de Janeiro |

**Details of 10250**

Order ID

Customer ID

Employee ID

Ship City

Argentina ▼

Save

Cancel

|       |       |                |
|-------|-------|----------------|
| 10251 | VICTE | Lyon           |
| 10252 | SUPRD | Charleroi      |
| 10253 | HANAR | Rio de Janeiro |
| 10254 | CHOPS | Bern           |
| 10255 | RICSU | Genève         |
| 10256 | WELLI | Resende        |
| 10257 | HILAA | San Cristóbal  |
| 10258 | ERNSH | Graz           |

Before the template elements are converted to JS controls

| OrderID | CustomerID | ShipCity       |
|---------|------------|----------------|
| 10248   | VINET      | Reims          |
| 10249   | TOMSP      | Münster        |
| 10250   | HANAR      | Rio de Janeiro |

**Details of 10250**

Order ID

Customer ID

Employee ID

Ship City

Save

Cancel

|       |       |                |
|-------|-------|----------------|
| 10251 | VICTE | Lyon           |
| 10252 | SUPRD | Charleroi      |
| 10253 | HANAR | Rio de Janeiro |
| 10254 | CHOPS | Bern           |
| 10255 | RICSU | Genève         |
| 10256 | WELLI | Resende        |
| 10257 | HILAA | San Cristóbal  |
| 10258 | ERNSH | Graz           |

After the template elements are converted to JS controls using `actionComplete` event

#### Dialog

Set `editMode` as `dialog` to edit data using a dialog box, which displays the fields associated with the data record being edited.

The following code example describes the above behavior.

#### HTML

```
<div id="Grid"></div>
```

#### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
```

```
//The datasource "window.gridData" is referred from  
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'  
dataSource: window["gridData"],  
toolbarSettings: {  
  showToolBar: true,  
  toolbarItems: ["add", "edit", "delete", "update", "cancel"]  
},  
editSettings: {  
  allowEditing: true,  
  allowAdding: true,  
  allowDeleting: true,  
  editMode: "dialog"  
},  
allowPaging: true,  
columns: [  
  { field: "OrderID", isPrimaryKey: true },  
  { field: "CustomerID" },  
  { field: "Freight", editType: "numericedit" },  
  { field: "ShipCountry", editType: "dropdownedit" },  
  { field: "OrderDate", editType: "datepicker", format: "{0:dd/MM/yyyy}" }  
]  
});  
});  
}
```

The following output is displayed as a result of the above code example.

The screenshot shows a grid with columns: OrderID, CustomerID, Freight, ShipCountry, and OrderDate. A dialog box titled "Details of 10250" is open, allowing editing of the selected record. The dialog contains input fields for OrderID (10250), CustomerID (HANAR), Freight (66), ShipCountry (Brazil), and OrderDate (08/07/1996). There are "Save" and "Cancel" buttons at the bottom of the dialog. The grid shows records 10248 through 10259. The bottom of the grid indicates "1 of 17 pages (200 items)".

| OrderID | CustomerID | Freight | ShipCountry | OrderDate  |
|---------|------------|---------|-------------|------------|
| 10248   | VINET      | 32.38   | France      | 04/07/1996 |
| 10249   | TOMSP      |         |             | 05/07/1996 |
| 10250   | HANAR      |         |             | 08/07/1996 |
| 10251   | VICTE      |         |             | 08/07/1996 |
| 10252   | SUPRD      |         |             | 09/07/1996 |
| 10253   | HANAR      |         |             | 10/07/1996 |
| 10254   | CHOPS      |         |             | 11/07/1996 |
| 10255   | RICSU      |         |             | 12/07/1996 |
| 10256   | WELLI      |         |             | 15/07/1996 |
| 10257   | HILAA      |         |             | 16/07/1996 |
| 10258   | ERNSH      |         |             | 17/07/1996 |
| 10259   | CENTC      | 3.25    | Mexico      | 18/07/1996 |

### Dialog Template Form

You can edit any of the fields pertaining to a single record of data and apply it to a template so that the same format is applied to all the other records that you may edit later.

Using this template support, you can edit the fields that are not bound to grid columns.

To edit the records using Inline template form, set [editMode](#) as dialogtemplate and specify the template id to [dialogEditorTemplateID](#) property of [editSettings](#).

While using template, you can change the elements that are defined in the [template](#), to appropriate JS controls based on the column type. This can be achieved by using [actionComplete](#) event of grid.

**Note:** 1. [value](#) attribute is used to bind the corresponding field value while editing.

2. [name](#) attribute is used to get the changed field values while save the edited record.

3. For [editMode](#) property you can assign either [string](#) value ("dialogtemplate") or [enum](#) value (ej.Grid.EditMode.DialogTemplate).

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
<script id="template" type="text/template">
<table cellpadding="10">
<tr>
```



```

<td>Order ID</td>
<td>
<input id="OrderID" name="OrderID" disabled="disabled"
value="{{{"{":{}:OrderID{}}}}}" class="e-field e-ejinputtext"
style="width:116px;height:28px" />
</td>
<td>Customer ID</td>
<td>
<input id="CustomerID" name="CustomerID" value="{{{"{":{}:CustomerID{}}}}}"
class="e-field e-ejinputtext" style="width: 116px; height: 28px" />
</td>
</tr>
<tr>
<td>Employee ID</td>
<td>
<input type="text" id="EmployeeID" name="EmployeeID"
value="{{{"{":{}:EmployeeID{}}}}}" />
</td>
<td>Ship City</td>
<td>
<select id="ShipCity" name="ShipCity">
<option value="Argentina">Argentina</option>
<option value="Austria">Austria</option>
<option value="Belgium">Belgium</option>
<option value="Brazil">Brazil</option>
<option value="Canada">Canada</option>
<option value="Denmark">Denmark</option>
</select>
</td>
</tr>
</table>
</script>

```

## TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
toolbarSettings: {
showToolbar: true,
toolbarItems: ["add", "edit", "delete", "update", "cancel"]
},
editSettings: {
allowEditing: true,
allowAdding: true,
allowDeleting: true,
editMode: "dialogtemplate",
dialogEditorTemplateID: "#template"
},
allowPaging: true,
columns: [

```

```

{ field: "OrderID", isPrimaryKey: true },
{ field: "CustomerID" },
{ field: "ShipCity" }
],
actionComplete: complete
});
});
}
function complete(args: ej.Grid.ActionCompleteEventArgs) {
$("#EmployeeID").ejNumericTextbox();
$("#Freight").ejNumericTextbox();
$("#ShipCity").ejDropDownList();
}

```

The following output is displayed as a result of the above code example.

The screenshot displays a grid with three columns: OrderID, CustomerID, and ShipCity. The grid contains 17 items. A modal dialog titled "Details of 10250" is open, showing the details for the selected row (OrderID 10250). The dialog includes input fields for Order ID (10250), Customer ID (HANAR), Employee ID (4), and Ship City (Argentina). There are Save and Cancel buttons at the bottom of the dialog. The grid's footer shows "1 of 17 pages (200 items)".

| OrderID | CustomerID | ShipCity       |
|---------|------------|----------------|
| 10248   | VINET      | Reims          |
| 10249   | TOMSP      | Münster        |
| 10250   | HANAR      | Rio de Janeiro |
| 10251   |            |                |
| 10252   |            |                |
| 10253   |            |                |
| 10254   |            |                |
| 10255   |            |                |
| 10256   |            |                |
| 10257   | HILAA      | San Cristóbal  |
| 10258   | ERNSH      | Graz           |
| 10259   | CENTC      | México D.F.    |

Before the template elements are converted to JS controls

The screenshot shows a grid with columns OrderID, CustomerID, and ShipCity. The row for OrderID 10250 is selected. An edit modal titled 'Details of 10250' is open, displaying the current values: Order ID (10250), Customer ID (HANAR), Employee ID (4), and Ship City (Belgium). The modal has 'Save' and 'Cancel' buttons. The grid footer shows '1 of 17 pages (200 items)'.

| OrderID | CustomerID | ShipCity       |
|---------|------------|----------------|
| 10248   | VINET      | Reims          |
| 10249   | TOMSP      | Münster        |
| 10250   | HANAR      | Rio de Janeiro |
| 10251   |            |                |
| 10252   |            |                |
| 10253   |            |                |
| 10254   |            |                |
| 10255   |            |                |
| 10256   |            |                |
| 10257   | HILAA      | San Cristóbal  |
| 10258   | ERNSH      | Graz           |
| 10259   | CENTC      | México D.F.    |

After the template elements are converted to JS controls using `actionComplete` event

#### External Form

By setting the `editMode` as `externalform`, the edit form is opened outside the grid content.

The following code example describes the above behavior.

#### HTML

```
<div id="Grid"></div>
```

#### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
toolbarSettings: {
showToolBar: true,
toolbarItems: ["add", "edit", "delete", "update", "cancel"]
},
editSettings: {
```

```
allowEditing: true,
allowAdding: true,
allowDeleting: true,
editMode: "externalform"
},
allowPaging: true,
pageSettings: {
  pageSize: 7
},
columns: [
  { field: "OrderID", isPrimaryKey: true },
  { field: "CustomerID" },
  { field: "Freight", editType: "numericedit" },
  { field: "ShipCountry", editType: "dropdownedit" },
  { field: "OrderDate", editType: "datepicker", format: "{0:dd/MM/yyyy}" }
]
});
});
}
```

The following output is displayed as a result of the above code example.

| OrderID | CustomerID | Freight | ShipCountry | OrderDate  |
|---------|------------|---------|-------------|------------|
| 10248   | VINET      | 32.38   | France      | 04/07/1996 |
| 10249   | TOMSP      | 11.61   | Germany     | 05/07/1996 |
| 10250   | HANAR      | 65.83   | Brazil      | 08/07/1996 |
| 10251   | VICTE      | 41.34   | France      | 08/07/1996 |
| 10252   | SUPRD      | 51.3    | Belgium     | 09/07/1996 |
| 10253   | HANAR      | 58.17   | Brazil      | 10/07/1996 |
| 10254   | CHOPS      | 22.98   | Switzerland | 11/07/1996 |

1 of 29 pages (200 items)

### Details of 10250

OrderID: 
 CustomerID:

Freight: 
 ShipCountry:

OrderDate:

Form Position:

You can position an external edit form in the following two ways.

1. Top-right
2. Bottom left

This can be achieved by setting the [formPosition](#) property of [editSettings](#) as "topright" or "bottomleft".

The following code example describes the above behavior.

#### HTML

```
<div id="Grid"></div>
```

#### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
```

```

$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
toolbarSettings: {
showToolbar: true,
toolbarItems: ["add", "edit", "delete", "update", "cancel"]
},
editSettings: {
allowEditing: true,
allowAdding: true,
allowDeleting: true,
editMode: "externalform",
formPosition: "topright"
},
allowPaging: true,
columns: [
{ field: "OrderID", isPrimaryKey: true },
{ field: "CustomerID" },
{ field: "Freight", editType: "numericedit" },
{ field: "ShipCountry", editType: "dropdownedit" }
]
});
});
}

```

The following output is displayed as a result of the above code example.

| OrderID | CustomerID | Freight | ShipCountry |
|---------|------------|---------|-------------|
| 10248   | VINET      | 32.38   | France      |
| 10249   | TOMSP      | 11.61   | Germany     |
| 10250   | HANAR      | 65.83   | Brazil      |
| 10251   | VICTE      | 41.34   | France      |
| 10252   | SUPRD      | 51.3    | Belgium     |
| 10253   | HANAR      | 58.17   | Brazil      |
| 10254   | CHOPS      | 22.98   | Switzerland |
| 10255   | RICSU      | 148.33  | Switzerland |
| 10256   | WELLI      | 13.97   | Brazil      |
| 10257   | HILAA      | 81.91   | Venezuela   |
| 10258   | ERNSH      | 140.51  | Austria     |
| 10259   | CENTC      | 3.25    | Mexico      |

Details of 10251

OrderID

10251

CustomerID

VICTE

Freight

41

ShipCountry

France

Save

Cancel

1 of 17 pages (200 items)

### External Template Form

You can edit any of the fields pertaining to a single record of data and apply it to a template so that the same format is applied to all the other records that you may edit later.

Using this template support, you can edit the fields that are not bound to grid columns.

To edit the records using External template form, set [editMode](#) as externalformtemplate and specify the template id to [externalFormTemplateID](#) property of [editSettings](#).

While using template, you can change the elements that are defined in the template, to appropriate JS controls based on the column type. This can be achieved by using [actionComplete](#) event of grid.

**Note:** 1. value attribute is used to bind the corresponding field value while editing.

2. **name** attribute is used to get the changed field values while save the edited record.

3. For `editMode` property you can assign either `string` value ("externalformtemplate") or `enum` value (ej.Grid.EditMode.ExternalFormTemplate).

The following code example describes the above behavior.

## HTML

```
<div id="Grid"></div>
<script id="template" type="text/template">
<table cellpadding="10">
<tr>
<td>Order ID</td>
<td>
<input id="OrderID" name="OrderID" disabled="disabled"
value="{{{"{":":OrderID{}}}" class="e-field e-einputtext"
style="width:116px;height:28px" />
</td>
<td>Customer ID</td>
<td>
<input id="CustomerID" name="CustomerID" value="{{{"{":":CustomerID{}}}"
class="e-field e-einputtext" style="width: 116px; height: 28px" />
</td>
</tr>
<tr>
<td>Employee ID</td>
<td>
<input type="text" id="EmployeeID" name="EmployeeID"
value="{{{"{":":EmployeeID{}}}" />
</td>
<td>Ship City</td>
<td>
<select id="ShipCity" name="ShipCity">
<option value="Argentina">Argentina</option>
<option value="Austria">Austria</option>
<option value="Belgium">Belgium</option>
<option value="Brazil">Brazil</option>
<option value="Canada">Canada</option>
<option value="Denmark">Denmark</option>
</select>
</td>
</tr>
</table>
</script>
```

TS

```
/// <reference path="tsfiles/jquery.d.ts" />
```

```
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
toolbarSettings: {
showToolbar: true,
toolbarItems: ["add", "edit", "delete", "update", "cancel"]
},
editSettings: {
allowEditing: true,
allowAdding: true,
allowDeleting: true,
editMode: "externalformtemplate",
externalFormTemplateID: "#template"
},
allowPaging: true,
pageSettings: {
pageSize: 5
},
columns: [
{ field: "OrderID", isPrimaryKey: true },
{ field: "CustomerID" },
{ field: "ShipCity" }
],
actionComplete: complete
});
});
}

function complete(args: ej.Grid.ActionCompleteEventArgs) {
$("#EmployeeID").ejNumericTextbox();
$("#Freight").ejNumericTextbox();
$("#ShipCity").ejDropDownList();
}
```

The following output is displayed as a result of the above code example.



| OrderID | CustomerID | ShipCity       |
|---------|------------|----------------|
| 10248   | VINET      | Reims          |
| 10249   | TOMSP      | Münster        |
| 10250   | HANAR      | Rio de Janeiro |
| 10251   | VICTE      | Lyon           |
| 10252   | SUPRD      | Charleroi      |

1

2

3

4

5

6

7

8

...

1 of 40 pages (200 items)

Details of 10250

Order ID

10250

Customer ID

HANAR

Employee ID

4

Ship City

Argentina ▼

Save

Cancel

Before the template elements are converted to JS controls

| OrderID | CustomerID | ShipCity       |
|---------|------------|----------------|
| 10248   | VINET      | Reims          |
| 10249   | TOMSP      | Münster        |
| 10250   | HANAR      | Rio de Janeiro |
| 10251   | VICTE      | Lyon           |
| 10252   | SUPRD      | Charleroi      |

1 of 40 pages (200 items)

### Details of 10250

Order ID:  Customer ID:

Employee ID:  Ship City:

After the template elements are converted to JS controls using `actionComplete` event

#### Batch / Excel-like

Users can start editing by clicking a cell and typing data into it. Edited cell will be marked while navigating to next cell or any other row, so that you know which fields or cells has been edited. Set `editMode` as `batch` to enable batch editing.

**Note:** Refer the KB [link](#) for "How to suppress grid confirmation messages" in batch mode.

The following code example describes the above behavior.

#### HTML

```
<div id="Grid"></div>
```

#### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
toolbarSettings: {
showToolbar: true,
toolbarItems: ["add", "edit", "delete", "update", "cancel"]
}
```

```

},
editSettings: {
allowEditing: true,
allowAdding: true,
allowDeleting: true,
editMode: "batch"
},
allowPaging: true,
columns: [
{ field: "OrderID", isPrimaryKey: true },
{ field: "CustomerID" },
{ field: "Freight", editType: "numericedit" },
{ field: "ShipCountry", editType: "dropdownedit" },
{ field: "OrderDate", editType: "datepicker", format: "{0:dd/MM/yyyy}" }
]
});
});
}

```

The following output is displayed as a result of the above code example.

| OrderID | CustomerID | Freight | ShipCountry | OrderDate  |
|---------|------------|---------|-------------|------------|
| 10248   | VINET      | 32.38   | France      | 04/07/1996 |
| 10249   | TOMSP      | 11.61   | Germany     | 05/07/1996 |
| 10250   | CHOPS      | 65.83   | Brazil      | 08/07/1996 |
| 10251   | CENTC      | 41.34   | France      | 08/07/1996 |
| 10252   | SUPRD      | 51.3    | Belgium     | 09/07/1996 |
| 10253   | HANAR      | 58.17   | Belgium     | 10/07/1996 |
| 10254   | CHOPS      | 22.98   | Switzerland | 11/07/1996 |
| 10255   | RICSU      | 150     | Switzerland | 12/07/1996 |
| 10256   | WELLI      | 13.97   | Brazil      | 09/09/2015 |
| 10257   | HILAA      | 81.91   | Venezuela   | 24/07/1996 |
| 10258   | ERNSH      | 140.51  | Austria     | 17/07/1996 |
| 10259   | CENTC      | 3.25    | Mexico      | 18/07/1996 |

1 of 17 pages (200 items)

### Confirmation messages

To show the confirm dialog while saving or discarding the batch changes (discarding during the grid action like filtering, sorting and paging), set [showConfirmDialog](#) as `true`.

---

**Note:** [showConfirmDialog](#) property is only for batch editing mode.

---

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.gridData" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            toolbarSettings: {
                showToolbar: true,
                toolbarItems: ["add", "edit", "delete", "update", "cancel"]
            },
            editSettings: {
                allowEditing: true,
                allowAdding: true,
                allowDeleting: true,
                editMode: "batch",
                showConfirmDialog: true
            },
            allowPaging: true,
            columns: [
                { field: "OrderID", isPrimaryKey: true },
                { field: "CustomerID" },
                { field: "Freight", editType: "numericedit" },
                { field: "ShipCountry", editType: "dropdownedit" },
                { field: "OrderDate", editType: "datepicker", format: "{0:dd/MM/yyyy}" }
            ]
        });
    });
}
```

The following output is displayed as a result of the above code example.

| OrderID | CustomerID | Freight | ShipCountry | OrderDate  |
|---------|------------|---------|-------------|------------|
| 10248   | VINET      | 32.38   | France      | 04/07/1996 |
| 10249   | TOMSP      | 11.61   | Germany     | 05/07/1996 |
| 10250   | ERNSH      | 65.83   | Argentina   | 08/07/1996 |
| 10251   | CENTC      | 41.34   | France      | 08/07/1996 |
| 10252   | SUPRD      | 51.3    | Belgium     | 17/07/1996 |
| 10253   | HANAR      |         |             | 10/07/1996 |
| 10254   | CHOPS      |         | Switzerland | 11/07/1996 |
| 10255   | RICSU      | 148.33  | Switzerland | 12/07/1996 |
| 10256   | RICSU      | 13.97   | Brazil      | 15/07/1996 |
| 10257   | HILAA      | 81.91   | Venezuela   | 16/07/1996 |
| 10258   | ERNSH      | 140.51  | Austria     | 17/07/1996 |
| 10259   | CENTC      | 3.25    | Mexico      | 18/07/1996 |

Are you sure you want to save changes?

OK Cancel

1 of 17 pages (200 items)

To show delete confirm dialog while deleting a record, set [showDeleteConfirmDialog](#) as true.

**Note:** [showDeleteConfirmDialog](#) property is for all type of [editMode](#).

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
toolbarSettings: {
showToolBar: true,
toolbarItems: ["add", "edit", "delete", "update", "cancel"]
},
editSettings: {
allowEditing: true,
```

```

allowAdding: true,
allowDeleting: true,
showDeleteConfirmDialog: true
},
allowPaging: true,
columns: [
{ field: "OrderID", isPrimaryKey: true },
{ field: "CustomerID" },
{ field: "Freight", editType: "numericedit" },
{ field: "ShipCountry", editType: "dropdownedit" },
{ field: "OrderDate", editType: "datepicker", format: "{0:dd/MM/yyyy}" }
]
});
});
}

```

The following output is displayed as a result of the above code example.

| OrderID | CustomerID | Freight | ShipCountry | OrderDate  |
|---------|------------|---------|-------------|------------|
| 10248   | VINET      | 32.38   | France      | 04/07/1996 |
| 10249   | TOMSP      | 11.61   | Germany     | 05/07/1996 |
| 10250   | HANAR      | 65.83   | Brazil      | 08/07/1996 |
| 10251   | VICTE      | 41.34   | France      | 08/07/1996 |
| 10252   | SUPRD      | 51.3    | Belgium     | 09/07/1996 |
| 10253   | HANAR      |         |             | 10/07/1996 |
| 10254   | CHOPS      |         | Switzerland | 11/07/1996 |
| 10255   | RICSU      | 148.33  | Switzerland | 12/07/1996 |
| 10256   | WELLI      | 13.97   | Brazil      | 15/07/1996 |
| 10257   | HILAA      | 81.91   | Venezuela   | 16/07/1996 |
| 10258   | ERNSH      | 140.51  | Austria     | 17/07/1996 |
| 10259   | CENTC      | 3.25    | Mexico      | 18/07/1996 |

Are you sure you want to Delete Record?

OKCancel

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

### Column Validation

We can validate the value of the added or edited record cell before saving.

The below validation script files are needed when editing is enabled with validation.

1. jquery.validate.min.js
2. jquery.validate.unobtrusive.min.js

*jQuery Validation*

You can set validation rules using [validationRules](#) property of [columns](#). The following are jQuery validation methods.

**List of JQuery validation methods**

| Rules      | Description  |
|------------|--|
| required   | Requires an element.                                   |
| remote     | Requests a resource to check the element for validity. |
| minlength  | Requires the element to be of given minimum length.    |
| maxlength  | Requires the element to be of given maximum length.    |
| range      | Requires the element to be in given value range.       |
| min        | The element requires a given minimum.                  |
| max        | The element requires a given maximum.                  |
| range      | Requires the element to be in a given value range.     |
| email      | The element requires a valid email.                    |
| url        | The element requires a valid URL                       |
| date       | Requires the element to be a date.                     |
| dateISO    | The element requires an ISO date.                      |
| number     | The element requires a decimal number.                 |
| digits     | The element requires digits only.                      |
| creditcard | Requires the element to be a credit card number.       |
| equalTo    | Requires the element to be the same as another.        |

Grid supports all the standard validation methods of jQuery, please refer to the jQuery validation documentation [link](#) for more information.

The following code example describes the above behavior.

**HTML**

```
<div id="Grid"></div>
```

**TS**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
```

```
//The datasource "window.gridData" is referred from  
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'  
dataSource: window["gridData"],  
toolbarSettings: {  
  showToolbar: true,  
  toolbarItems: ["add", "edit", "delete", "update", "cancel"]  
},  
editSettings: {  
  allowEditing: true,  
  allowAdding: true,  
  allowDeleting: true,  
  showDeleteConfirmDialog: true  
},  
allowPaging: true,  
columns: [  
  { field: "OrderID", isPrimaryKey: true, validationRules: { required: true,  
    number: true } },  
  { field: "CustomerID", validationRules: { required: true, minlength: 3 } },  
  { field: "ShipCity" },  
  { field: "Freight", editType: "numericedit", validationRules: { range: [0,  
    1000] } },  
  { field: "ShipCountry" }  
]  
});  
});  
}
```

The following output is displayed as a result of the above code example.



| OrderID              | CustomerID                          | ShipCity       | Freight                                  | ShipCountry |
|----------------------|-------------------------------------|----------------|--|-------------|
|                      | VI                                  |                | 4,567                                    |             |
| OrderID is required. | Please enter at least 3 characters. | Reims          | Please enter a value between 0 and 1000. | France      |
| 10249                | TOUSP                               | Münster        |  | Germany     |
| 10250                | HANAR                               | Rio de Janeiro | 65.83                                    | Brazil      |
| 10251                | VICTE                               | Lyon           | 41.34                                    | France      |
| 10252                | SUPRD                               | Charleroi      | 51.3                                     | Belgium     |
| 10253                | HANAR                               | Rio de Janeiro | 58.17                                    | Brazil      |
| 10254                | CHOPS                               | Bern           | 22.98                                    | Switzerland |
| 10255                | RICSU                               | Genève         | 148.33                                   | Switzerland |
| 10256                | WELLI                               | Resende        | 13.97                                    | Brazil      |
| 10257                | HILAA                               | San Cristóbal  | 81.91                                    | Venezuela   |
| 10258                | ERNSH                               | Graz           | 140.51                                   | Austria     |

1 of 17 pages (201 items)

**Note:** 1. Refer this [Knowledge Base link](#) to perform server side validation in Grid.

### Custom Validation

In addition to jQuery validation methods, you can also add your own custom validation methods for a specific column. Function call to custom validator function to be mentioned within [validationRules](#) property of [columns](#).

Using [messages](#) property of [validationRules](#) you can specify the error message for that column.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/jquery.validation.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        $.validator.addMethod("customCompare", function (value: any, element:
        HTMLInputElement, params: any) {
            return element.value > params[0] && element.value < params[1]
        }, "Freight value must be between 0 and 1000");
    });
}
```

```
$.validator.addMethod("customRegex", function (value, element:
HTMLInputElement, params) {
    if (element.value.length == params)
        return true;
    return false;
}, "Customer ID must be 5 characters");
var gridInstance = new ej.Grid($("#Grid"), {
    //The datasource "window.gridData" is referred from
    'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
    dataSource: window["gridData"],
    toolbarSettings: {
        showToolBar: true,
        toolbarItems: ["add", "edit", "delete", "update", "cancel"]
    },
    editSettings: {
        allowEditing: true,
        allowAdding: true,
        allowDeleting: true,
        showDeleteConfirmDialog: true
    },
    allowPaging: true,
    columns: [
        { field: "OrderID", isPrimaryKey: true, validationRules: { required: true,
            number: true } },
        { field: "CustomerID", validationRules: { customRegex: 5 } },
        { field: "ShipCity" },
        { field: "Freight", editType: "numericedit", validationRules: {
            customCompare: [0, 1000] } },
        { field: "ShipCountry" }
    ]
});
});
}
```

The following output is displayed as a result of the above code example.

| OrderID              | CustomerID                       | ShipCity       | Freight                                  | ShipCountry |
|----------------------|----------------------------------|----------------|--|-------------|
|                      |                                  |                | Enter value                              |             |
| OrderID is required. | Customer ID must be 5 characters | Reims          | Freight value must be between 0 and 1000 | France      |
| 10249                | TOUSP                            | Münster        |  | Germany     |
| 10250                | HANAR                            | Rio de Janeiro | 65.83                                    | Brazil      |
| 10251                | VICTE                            | Lyon           | 41.34                                    | France      |
| 10252                | SUPRD                            | Charleroi      | 51.3                                     | Belgium     |
| 10253                | HANAR                            | Rio de Janeiro | 58.17                                    | Brazil      |
| 10254                | CHOPS                            | Bern           | 22.98                                    | Switzerland |
| 10255                | RICSU                            | Genève         | 148.33                                   | Switzerland |
| 10256                | WELLI                            | Resende        | 13.97                                    | Brazil      |
| 10257                | HILAA                            | San Cristóbal  | 81.91                                    | Venezuela   |
| 10258                | ERNSH                            | Graz           | 140.51                                   | Austria     |

1 of 17 pages (201 items)

### Persisting data in Server

Edited data can be persisted in database using RESTful web services.

All the CRUD operations in grid are done through DataManager. DataManager have an option to bind all the CRUD related data in server side. Please refer to the ['link'](#) to know about the DataManager.

For you information ODataAdaptor persist data in server as per OData protocol.

In the below section, we have explained how to get the edited data details at the server side using URLAdaptor.

#### URL Adaptor

You can use the `UrlAdaptor` of `ejDataManager` when binding datasource from remote data. At initial load of Grid, using URL property of DataManager, data are fetched from remote data and bound to Grid. You can map CRUD operation in Grid to Server-Side Controller action using the properties `insertUrl`, `removeUrl`, `updateUrl`, `crudUrl` and `batchUrl`.

The following code example describes the above behavior.

#### HTML

```
<div id="Grid"></div>
```

#### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
```

```

/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
dataSource: new ej.DataManager({
url: "Home/DataSource",
updateUrl: "Home/Update",
insertUrl: "Home/Insert",
removeUrl: "Home/Delete",
adaptor: "UrlAdaptor"
}),
allowPaging: true,
editSettings: {
allowEditing: true,
allowAdding: true,
allowDeleting: true
},
toolbarSettings: {
showToolbar: true,
toolbarItems: [ej.Grid.ToolBarItems.Add, ej.Grid.ToolBarItems.Edit,
ej.Grid.ToolBarItems.Delete, ej.Grid.ToolBarItems.Update,
ej.Grid.ToolBarItems.Cancel]
},
columns: [
{ field: "OrderID", isPrimaryKey: true },
{ field: "CustomerID" },
{ field: "EmployeeID" },
{ field: "Freight", editType: ej.Grid.EditingType.Numeric, editParams: {
decimalPlaces: 2 }, format: "{0:C}" },
{ field: "ShipName" },
{ field: "ShipCountry" }
]
});
});
}

```

Also when you use `UrlAdaptor`, you need to return the data as `JSON` and the `JSON` object must contain a property as `result` with the `dataSource` as its value and one more property `count` with the `dataSource` total records count as its value.

The following code example describes the above behavior.

### C#

```

public ActionResult DataSource(DataManager dataManager)
{
IEnumerable DataSource = OrderRepository.GetAllRecords();
DataResult result = new DataResult();
DataOperations operation = new DataOperations();
result.result = DataSource;
result.count = result.result.AsQueryable().Count();
if (dataManager.Skip > 0)
result.result = operation.PerformSkip(result.result, dataManager.Skip);
if (dataManager.Take > 0)
result.result = operation.PerformTake(result.result, dataManager.Take);
return Json(result, JsonRequestBehavior.AllowGet);
}

```

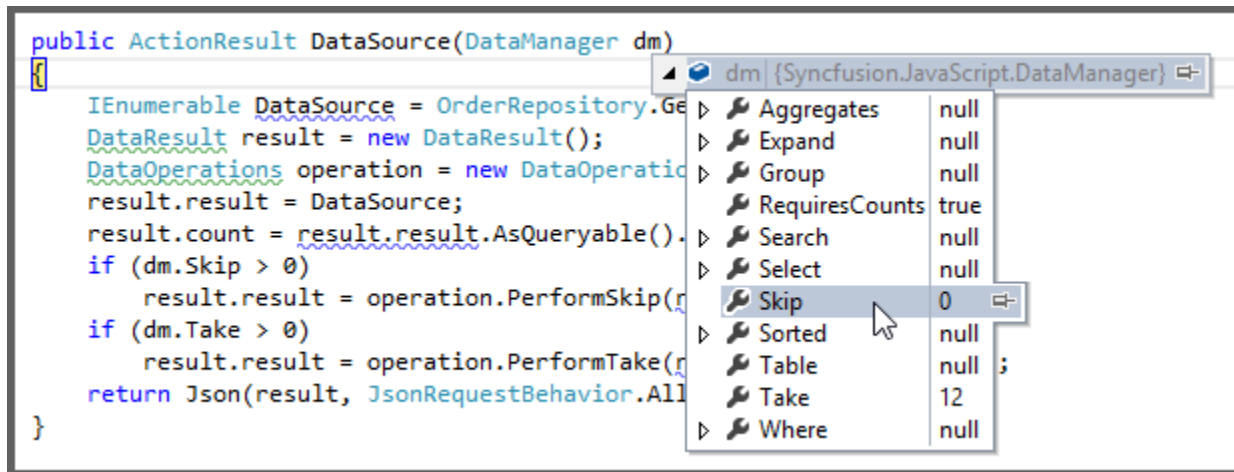
```

}
public class DataResult
{
public IEnumerable result { get; set; }
public int count { get; set; }
}

```

The grid actions (sorting, filtering, paging, searching, and aggregates) details are obtained in the 'DataManager' class. While initializing the grid, paging only enabled hence in the below screen shot paging details are bound to the DataManager class.

Please refer the below screen shot.



Also, using 'DataOperations' helper class you can perform grid action at server side. The built-in methods that we have provided in the DataOperations class are listed below.

1. PerformSorting
2. PerformFiltering
3. PerformSearching
4. PerformSkip
5. PerformTake
6. PerformWhereFilter
7. PerformSelect
8. Execute

*Accessing CRUD action request details in server side:*

The 'Server-Side' function must be declared with the following parameter name for each editing functionality.

Parameters Table

| Action        | Parameter Name  | Example   |
|---------------|---|---|
| Update,Insert | value   | public ActionResult<br>Update(EditableOrder value){ } |
|               | public ActionResult<br>Insert(EditableOrder value){ } |   |

|                            |                        |   |
|----------------------------|------------------------|---|
| Remove                     | key                    | public ActionResult Remove(int key){ }  |
| Batch Add                  | added                  | public ActionResult BatchUpdate(string action, List <EditableOrder> added, List <EditableOrder> changed, List <EditableOrder> deleted, int? key){ } |
| Batch Update               | changed                |   |
| Batch Delete               | deleted                |   |
| Crud Update, Crud Insert   | value, action          | public ActionResult CrudUrl(EditableOrder value, string action){ }  |
| Crud Remove                | action, key, keyColumn | public ActionResult CrudUrl(string action, int? key, string keyColumn){ }   |
| Crud Remove - Multi Delete | action, key, deleted   | public ActionResult CrudUrl(string action, string key, List <EditableOrder> deleted){ }   |

#### Insert Record:

Using `insertUrl` property, you can specify the controller action mapping URL to perform insert operation at the server side.

The following code example describes the above behavior.

#### C#

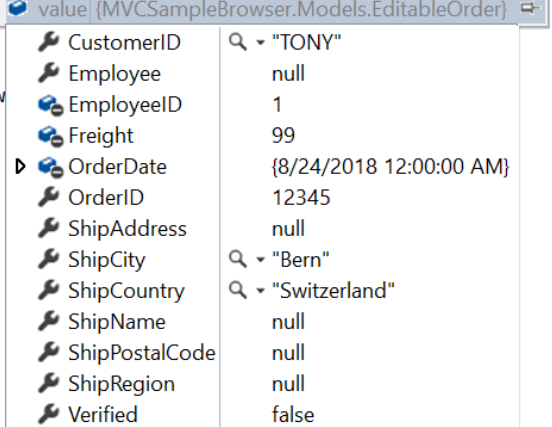
```
public ActionResult Insert(EditableOrder value)
{
    OrderRepository.Add(value);
    var data = OrderRepository.GetAllRecords();
    return Json(value, JsonRequestBehavior.AllowGet);
}
```

The newly added record details are bound to the 'value' parameter. Please refer to the below image.

```

0 references
public ActionResult Insert(EditableOrder value)
{
    < 41,827ms elapsed
    OrderRepository.Add(value);
    var data = OrderRepository.GetAllRecords();
    return Json(value, JsonRequestBehavior.AllowGet);
}

```



| Property       | Value                   |
|----------------|-------------------------|
| CustomerID     | TONY                    |
| Employee       | null                    |
| EmployeeID     | 1                       |
| Freight        | 99                      |
| OrderDate      | {8/24/2018 12:00:00 AM} |
| OrderID        | 12345                   |
| ShipAddress    | null                    |
| ShipCity       | Bern                    |
| ShipCountry    | Switzerland             |
| ShipName       | null                    |
| ShipPostalCode | null                    |
| ShipRegion     | null                    |
| Verified       | false                   |

#### Update Record:

Using `updateUrl` property, you can specify the controller action mapping URL to perform save/update operation at server side.

The following code example describes the above behavior.

#### C#

```

public ActionResult Update(EditableOrder value)
{
    OrderRepository.Update(value);
    var data = OrderRepository.GetAllRecords();
    return Json(value, JsonRequestBehavior.AllowGet);
}

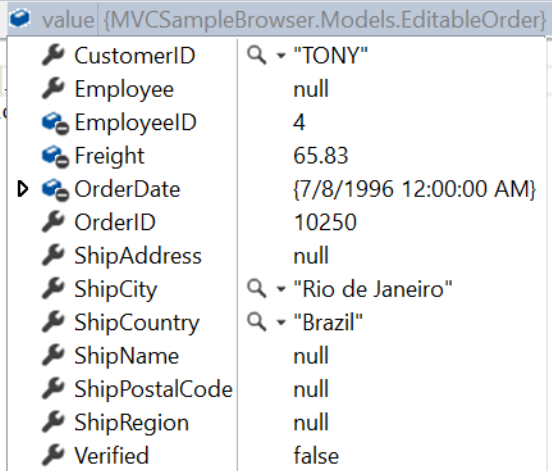
```

The updated record details are bound to the 'value' parameter. Please refer to the below image.

```

0 references
public ActionResult Update(EditableOrder value)
{
    < 5,620ms elapsed
    OrderRepository.Update(value);
    var data = OrderRepository.GetAllRecords();
    return Json(value, JsonRequestBehavior.AllowGet);
}

```



| Property       | Value                  |
|----------------|------------------------|
| CustomerID     | TONY                   |
| Employee       | null                   |
| EmployeeID     | 4                      |
| Freight        | 65.83                  |
| OrderDate      | {7/8/1996 12:00:00 AM} |
| OrderID        | 10250                  |
| ShipAddress    | null                   |
| ShipCity       | Rio de Janeiro         |
| ShipCountry    | Brazil                 |
| ShipName       | null                   |
| ShipPostalCode | null                   |
| ShipRegion     | null                   |
| Verified       | false                  |

*Delete Record:*

Using `removeUrl` property, you can specify the controller action mapping URL to perform delete operation at the server side.

The following code example describes the above behavior.

**C#**

```
public ActionResult Remove(int key)
{
    OrderRepository.Delete(key);
    var data = OrderRepository.GetAllRecords();
    return Json(key, JsonRequestBehavior.AllowGet);
}
```

The deleted record primary key value is bound to the 'key' parameter. Please refer the below image.

0 references

```
public ActionResult Remove(int key)
{
    OrderRepository.Delete(key);
    var data = OrderRepository.GetAllRecords();
    return Json(key, JsonRequestBehavior.AllowGet);
}
```

≤ 9,693ms elapsed

key 10248

*CRUD URL:*

Instead of specifying separate controller action method for CRUD (insert, update and delete) operation, using `crudUrl` property you can specify the controller action mapping URL to perform all CRUD operation at the server side using single method.

The action parameter of `crudUrl` is used to get the corresponding CRUD action.

The following code example describes the above behavior.

**HTML**

```
<div id="Grid"></div>
```

**TS**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            dataSource: new ej.DataManager({
                url: "Home/DataSource",
                crudUrl: "Home/CrudUpdate",
                adaptor: "UrlAdaptor"
            }),
            allowPaging: true,
            editSettings: {
                allowEditing: true,
                allowAdding: true,
                allowDeleting: true
            }
        });
    });
}
```



```

},
toolbarSettings: {
  showToolBar: true,
  toolbarItems: [ej.Grid.ToolBarItems.Add, ej.Grid.ToolBarItems.Edit,
ej.Grid.ToolBarItems.Delete, ej.Grid.ToolBarItems.Update,
ej.Grid.ToolBarItems.Cancel]
},
columns: [
  { field: "OrderID", isPrimaryKey: true },
  { field: "CustomerID" },
  { field: "EmployeeID" },
  { field: "Freight", editType: ej.Grid.EditingType.Numeric, editParams: {
decimalPlaces: 2 }, format: "{0:C}" },
  { field: "ShipName" },
  { field: "ShipCountry" }
]
});
});
}

```

## JAVASCRIPT

```

public ActionResult CrudUpdate(EditableOrder value, string action, int key)
{
  if (action == "update")
    OrderRepository.Update(value);
  else if (action == "insert")
    OrderRepository.Add(value);
  else if (action == "remove")
    OrderRepository.Delete(key);
  return Json(value, JsonRequestBehavior.AllowGet);
}

```

Please refer to the below image to know about the action parameter

```

public ActionResult CrudUrl(EditableOrder value, int? key, string action)
{
  if (action == "update")
    OrderRepository.Update(value);
  else if (action == "insert")
    OrderRepository.Add(value);
  else if (action == "remove")
    OrderRepository.Delete(key);
  return Json(value, JsonRequestBehavior.AllowGet);
}

```



**Note:** If you specify `insertUrl` along with `CrudUrl` then while adding `insertUrl` only called.

### Batch URL:

The `batchUrl` property supports only for batch editing mode. You can specify the controller action mapping URL to perform Batch operation at the server side.

The following code example describes the above behavior.

## HTML

```
<div id="Grid"></div>
```

## TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
dataSource: new ej.DataManager({
url: "Home/DataSource",
batchUrl: "Home/BatchUpdate",
adaptor: "UrlAdaptor"
}),
allowPaging: true,
editSettings: {
allowEditing: true,
allowAdding: true,
allowDeleting: true,
editMode: "batch"
},
toolbarSettings: {
showToolBar: true,
toolbarItems: [ej.Grid.ToolBarItems.Add, ej.Grid.ToolBarItems.Edit,
ej.Grid.ToolBarItems.Delete, ej.Grid.ToolBarItems.Update,
ej.Grid.ToolBarItems.Cancel]
},
columns: [
{ field: "OrderID", isPrimaryKey: true },
{ field: "CustomerID" },
{ field: "EmployeeID" },
{ field: "Freight", editType: ej.Grid.EditingType.Numeric, editParams: {
decimalPlaces: 2 }, format: "{0:C}" },
{ field: "ShipName" },
{ field: "ShipCountry" }
]
});
});
}

```

## C#

```

public ActionResult BatchUpdate(string action, List<EditableOrder> added,
List<EditableOrder> changed, List<EditableOrder> deleted, int? key)
{
if (changed != null)
OrderRepository.Update(changed);
if (deleted != null)
OrderRepository.Delete(deleted);
if (added != null)
OrderRepository.Add(added);
var data = OrderRepository.GetComplexRecords();
return Json(new { changed = changed, added = added, deleted = deleted },
JsonRequestBehavior.AllowGet);
}

```

Please refer to the below image for more information about batch parameters

```

0 references
public ActionResult BatchUpdate(string action, List<EditableOrder> added, List<EditableOrder> changed, List<EditableOrder> deleted, int? key)
{
    < 103,839ms elapsed
    if (changed != null)
        OrderRepository.Update(changed);
    if (deleted != null)
        OrderRepository.Delete(deleted);
    if (added != null)
        OrderRepository.Add(added);
    var data = OrderRepository.GetComplexRecords();
    return Json(new { changed = changed, added = added, deleted = deleted }, JsonRequestBehavior.AllowGet);
}

```

| changed        | Count = 3                               |
|----------------|---|
| [0]            | {MVCSampleBrowser.Models.EditableOrder} |
| [1]            | {MVCSampleBrowser.Models.EditableOrder} |
| CustomerID     | "HANAR"                                 |
| Employee       | null                                    |
| EmployeeID     | 3                                       |
| Freight        | 245                                     |
| OrderDate      | {7/10/1996 12:00:00 AM}                 |
| OrderID        | 10253                                   |
| ShipAddress    | "Rua do Paço, 67"                       |
| ShipCity       | "Rio de Janeiro"                        |
| ShipCountry    | "Brazil"                                |
| ShipName       | "Hanari Carnes"                         |
| ShipPostalCode | "05454-876"                             |
| ShipRegion     | "RJ"                                    |
| Verified       | true                                    |

### Adding New Row Position

To add a new row in the top or bottom position of grid content, set [rowPosition](#) property of [editSettings](#) depending on the requirement.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.gridData" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            toolbarSettings: {
                showToolbar: true,
                toolbarItems: ["add", "edit", "delete", "update", "cancel"]
            },
            editSettings: {
                allowEditing: true,
                allowAdding: true,
                allowDeleting: true,
                rowPosition: "bottom"
            },
            allowPaging: true,
            columns: [
                { field: "OrderID", isPrimaryKey: true },
                { field: "CustomerID" },
                { field: "ShipCity" },
                { field: "Freight", editType: "numericedit" },
            ]
        });
    });
}

```

```
{ field: "ShipCountry" }
]
});
});
}
```

The following output is displayed as a result of the above code example.

| OrderID              | CustomerID           | ShipCity             | Freight                                  | ShipCountry          |
|----------------------|----------------------|----------------------|--|----------------------|
| 10248                | VINET                | Reims                | 32.38                                    | France               |
| 10249                | TOMSP                | Münster              | 11.61                                    | Germany              |
| 10250                | HANAR                | Rio de Janeiro       | 65.83                                    | Brazil               |
| 10251                | VICTE                | Lyon                 | 41.34                                    | France               |
| 10252                | SUPRD                | Charleroi            | 51.3                                     | Belgium              |
| 10253                | HANAR                | Rio de Janeiro       | 58.17                                    | Brazil               |
| 10254                | CHOPS                | Bern                 | 22.98                                    | Switzerland          |
| 10255                | RICSU                | Genève               | 148.33                                   | Switzerland          |
| 10256                | WELLI                | Resende              | 13.97                                    | Brazil               |
| 10257                | HILAA                | San Cristóbal        | 81.91                                    | Venezuela            |
| 10258                | ERNSH                | Graz                 | 140.51                                   | Austria              |
| <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text" value="Enter value"/> | <input type="text"/> |

1 of 17 pages (201 items)

Render with blank row for easy add new

The blank add new row is displayed in the grid content during grid initialization itself to add a new record easily. To enable show add new row by default, set the [showAddNewRow](#) property of [editSettings](#) as `true`.

The blank add new row is displayed either in the top or bottom of the corresponding page, its position is based on the [rowPosition](#) property of [editSettings](#).

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
///

```

```
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
toolbarSettings: {
showToolbar: true,
toolbarItems: ["add", "edit", "delete", "update", "cancel"]
},
editSettings: {
allowEditing: true,
allowAdding: true,
allowDeleting: true,
showAddNewRow: true
},
allowPaging: true,
columns: [
{ field: "OrderID", isPrimaryKey: true },
{ field: "CustomerID" },
{ field: "ShipCity" },
{ field: "Freight", editType: "numericedit" },
{ field: "ShipCountry" }
]
});
});
}
```

The following output is displayed as a result of the above code example.

| OrderID              | CustomerID           | ShipCity             | Freight                                  | ShipCountry          |
|----------------------|----------------------|----------------------|--|----------------------|
| <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text" value="Enter value"/> | <input type="text"/> |
| 10248                | VINET                | Reims                | 32.38                                    | France               |
| 10249                | TOMSP                | Münster              | 11.61                                    | Germany              |
| 10250                | HANAR                | Rio de Janeiro       | 65.83                                    | Brazil               |
| 10251                | VICTE                | Lyon                 | 41.34                                    | France               |
| 10252                | SUPRD                | Charleroi            | 51.3                                     | Belgium              |
| 10253                | HANAR                | Rio de Janeiro       | 58.17                                    | Brazil               |
| 10254                | CHOPS                | Bern                 | 22.98                                    | Switzerland          |
| 10255                | RICSU                | Genève               | 148.33                                   | Switzerland          |
| 10256                | WELLI                | Resende              | 13.97                                    | Brazil               |
| 10257                | HILAA                | San Cristóbal        | 81.91                                    | Venezuela            |
| 10258                | ERNSH                | Graz                 | 140.51                                   | Austria              |
| 10259                | CENTC                | México D.F.          | 3.25                                     | Mexico               |

1
2
3
4
5
6
7
8
...

1 of 17 pages (200 items)

**Note:** 1. If it is remote, then the newly added record is placed based on the index from current view data.

**Note:** 2. If it is local, then the newly added record is added at the top of the page even if the added new [rowPosition](#) is mentioned as "bottom".

#### Default column values on add new

While adding new record in grid, there is an option to set the default value for the columns. Using the [defaultValue](#) property of [columns](#) you can set the default values for that particular column while editing or adding a new row.

The following code example describes the above behavior.

#### HTML

```
<div id="Grid"></div>
```

#### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
```

```
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
toolbarSettings: {
  showToolbar: true,
  toolbarItems: ["add", "edit", "delete", "update", "cancel"]
},
editSettings: {
  allowEditing: true,
  allowAdding: true,
  allowDeleting: true
},
allowPaging: true,
columns: [
  { field: "OrderID", isPrimaryKey: true },
  { field: "CustomerID" },
  { field: "ShipCity", defaultValue: "Bern" },
  { field: "Freight", editType: "numericedit", defaultValue: 45 },
  { field: "ShipCountry", defaultValue: "Brazil" }
]
});
});
}
```

The following output is displayed as a result of the above code example.

| OrderID | CustomerID | ShipCity       | Freight | ShipCountry |
|---------|------------|----------------|---------|-------------|
|         |            | Bern           | 45      | Brazil      |
| 10248   | VINET      | Reims          | 32.38   | France      |
| 10249   | TOMSP      | Münster        | 11.61   | Germany     |
| 10250   | HANAR      | Rio de Janeiro | 65.83   | Brazil      |
| 10251   | VICTE      | Lyon           | 41.34   | France      |
| 10252   | SUPRD      | Charleroi      | 51.3    | Belgium     |
| 10253   | HANAR      | Rio de Janeiro | 58.17   | Brazil      |
| 10254   | CHOPS      | Bern           | 22.98   | Switzerland |
| 10255   | RICSU      | Genève         | 148.33  | Switzerland |
| 10256   | WELLI      | Resende        | 13.97   | Brazil      |
| 10257   | HILAA      | San Cristóbal  | 81.91   | Venezuela   |
| 10258   | ERNSH      | Graz           | 140.51  | Austria     |

1
2
3
4
5
6
7
8
...

1 of 17 pages (201 items)

## Filtering

Filtering helps to view particular or related records from dataSource which meets a given filtering criteria. To enable filter, set `allowFiltering` as `true`.

The Grid supports three types of filter, they are

1. Filter bar
2. Menu
3. Excel

And also four types of filter menu is available in all filter types, they are

1. String
2. Numeric
3. Date
4. Boolean

The corresponding filter menu is opened based on the column type.

**Note:** 1. Need to specify the [type](#) of column, when first record data value is empty or null otherwise the filter menu is not opened.

2. The default filter type is Filter bar, when `allowFiltering` is enabled and [filterType](#) is not set.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.gridData" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            allowPaging: true,
            allowFiltering: true,
            columns: ["OrderID", "EmployeeID", "CustomerID", "ShipCountry", "Freight"]
        });
    });
}
```

The following output is displayed as a result of the above code example.



| OrderID              | EmployeeID           | CustomerID           | ShipCountry          | Freight              |
|----------------------|----------------------|----------------------|----------------------|----------------------|
| <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> |
| 10248                | 5                    | VINET                | France               | 32.38                |
| 10249                | 6                    | TOMSP                | Germany              | 11.61                |
| 10250                | 4                    | HANAR                | Brazil               | 65.83                |
| 10251                | 3                    | VICTE                | France               | 41.34                |
| 10252                | 4                    | SUPRD                | Belgium              | 51.3                 |
| 10253                | 3                    | HANAR                | Brazil               | 58.17                |
| 10254                | 5                    | CHOPS                | Switzerland          | 22.98                |
| 10255                | 9                    | RICSU                | Switzerland          | 148.33               |
| 10256                | 3                    | WELLI                | Brazil               | 13.97                |
| 10257                | 4                    | HILAA                | Venezuela            | 81.91                |
| 10258                | 1                    | ERNSH                | Austria              | 140.51               |
| 10259                | 4                    | CENTC                | Mexico               | 3.25                 |

⏪

⏩

1

2

3

4

5

6

7

8

...

⏪

⏩

1 of 17 pages (200 items)

### Menu filter

You can enable menu filter by setting [filterSettings.filterType](#) as `menu`.

There is an option to show or hide the additional filter options in the menu by setting [filterSettings.showPredicate](#) as `true` or `false` respectively.

**Note:** For [filterType](#) property you can assign either `string` value ("menu") or `enum` value (ej.Grid.FilterType.Menu).

We can also filter a specified range of values by using the `between` operator for the column type `number`, `date` and `datetime`.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
```

```
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
allowPaging: true,
allowFiltering: true,
filterSettings: {
  filterType: "menu"
},
columns: [
  { field: "OrderID" },
  { field: "EmployeeID" },
  { field: "CustomerID" },
  { field: "OrderDate", format: "{0:dd/MM/yyyy}" },
  { field: "Verified" }
]
});
});
}
```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID | CustomerID | OrderDate  | Verified                            |
|---------|------------|------------|------------|-------------------------------------|
| 10248   | 5          |            |            | <input checked="" type="checkbox"/> |
| 10249   | 6          |            |            | <input type="checkbox"/>            |
| 10250   | 4          |            |            | <input checked="" type="checkbox"/> |
| 10251   | 3          |            |            | <input checked="" type="checkbox"/> |
| 10252   | 4          |            |            | <input checked="" type="checkbox"/> |
| 10253   | 3          |            |            | <input checked="" type="checkbox"/> |
| 10254   | 5          | CHOPS      | 11/07/1996 | <input type="checkbox"/>            |
| 10255   | 9          | RICSU      | 12/07/1996 | <input checked="" type="checkbox"/> |
| 10256   | 3          | WELLI      | 15/07/1996 | <input type="checkbox"/>            |
| 10257   | 4          | HILAA      | 16/07/1996 | <input checked="" type="checkbox"/> |
| 10258   | 1          | ERNSH      | 17/07/1996 | <input checked="" type="checkbox"/> |
| 10259   | 4          | CENTC      | 18/07/1996 | <input type="checkbox"/>            |

1 of 17 pages (200 items)

### Numeric Filter

| OrderID | EmployeeID | CustomerID | OrderDate  | Verified                            |
|---------|------------|------------|------------|-------------------------------------|
| 10248   | 5          | VINET      |            |                                     |
| 10249   | 6          | TOMSP      |            |                                     |
| 10250   | 4          | HANAR      |            |                                     |
| 10251   | 3          | VICTE      |            |                                     |
| 10252   | 4          | SUPRD      |            |                                     |
| 10253   | 3          | HANAR      | 10/07/1996 | <input checked="" type="checkbox"/> |
| 10254   | 5          | CHOPS      | 11/07/1996 | <input type="checkbox"/>            |
| 10255   | 9          | RICSU      | 12/07/1996 | <input checked="" type="checkbox"/> |
| 10256   | 3          | WELLI      | 15/07/1996 | <input type="checkbox"/>            |
| 10257   | 4          | HILAA      | 16/07/1996 | <input checked="" type="checkbox"/> |
| 10258   | 1          | ERNSH      | 17/07/1996 | <input checked="" type="checkbox"/> |
| 10259   | 4          | CENTC      | 18/07/1996 | <input type="checkbox"/>            |

1 of 17 pages (200 items)

## String Filter

| OrderID | EmployeeID | CustomerID | OrderDate  | Verified                            |
|---------|------------|------------|------------|-------------------------------------|
| 10248   | 5          | VINET      | 04/07/1996 |                                     |
| 10249   | 6          | TOMSP      | 05/07/1996 |                                     |
| 10250   | 4          | HANAR      | 08/07/1996 |                                     |
| 10251   | 3          | VICTE      | 08/07/1996 |                                     |
| 10252   | 4          | SUPRD      | 09/07/1996 |                                     |
| 10253   | 3          | HANAR      | 10/07/1996 |                                     |
| 10254   | 5          | CHOPS      | 11/07/1996 | <input type="checkbox"/>            |
| 10255   | 9          | RICSU      | 12/07/1996 | <input checked="" type="checkbox"/> |
| 10256   | 3          | WELLI      | 15/07/1996 | <input type="checkbox"/>            |
| 10257   | 4          | HILAA      | 16/07/1996 | <input checked="" type="checkbox"/> |
| 10258   | 1          | ERNSH      | 17/07/1996 | <input checked="" type="checkbox"/> |
| 10259   | 4          | CENTC      | 18/07/1996 | <input type="checkbox"/>            |

1 of 17 pages (200 items)

## Date Filter

| OrderID | EmployeeID | CustomerID | OrderDate  | Verified                            |
|---------|------------|------------|------------|-------------------------------------|
| 10248   | 5          | VINET      | 04/07/1996 | <input checked="" type="checkbox"/> |
| 10249   | 6          | TOMSP      | 05/07/1996 | <input type="checkbox"/>            |
| 10250   | 4          | HANAR      | 08/07/1996 | <input checked="" type="checkbox"/> |
| 10251   | 3          | VICTE      | 08/07/1996 | <input checked="" type="checkbox"/> |
| 10252   | 4          | SUPRD      | 09/07/1996 | <input checked="" type="checkbox"/> |
| 10253   | 3          | HANAR      | 10/07/1996 | <input checked="" type="checkbox"/> |
| 10254   | 5          | CHOPS      | 11/07/1996 | <input type="checkbox"/>            |
| 10255   | 9          | RICSU      | 12/07/1996 | <input checked="" type="checkbox"/> |
| 10256   | 3          | WELLI      | 15/07/1996 | <input type="checkbox"/>            |
| 10257   | 4          | HILAA      | 16/07/1996 | <input checked="" type="checkbox"/> |
| 10258   | 1          | ERNSH      | 17/07/1996 | <input checked="" type="checkbox"/> |
| 10259   | 4          | CENTC      | 18/07/1996 | <input type="checkbox"/>            |

Filter Value :

1 of 17 pages (200 items)

### Boolean Filter

### Excel-like filter

You can enable excel menu by setting `filterSettings.filterType` as `excel`. The excel menu contains an option such as Sorting, Clear filter, submenu for advanced filtering.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
allowPaging: true,
allowSorting: true,
allowFiltering: true,
filterSettings: {
filterType: "excel"
},
columns: ["OrderID", "EmployeeID", "CustomerID", "ShipCountry", "Freight"]
});
});
}

```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID | CustomerID | ShipCountry | Freight |
|---------|------------|------------|-------------|---------|
| 10248   | 5          |            |             | 32.38   |
| 10249   | 6          |            |             | 11.61   |
| 10250   | 4          |            |             | 65.83   |
| 10251   | 3          |            |             | 41.34   |
| 10252   | 4          |            |             | 51.3    |
| 10253   | 3          |            |             | 58.17   |
| 10254   | 5          |            |             | 22.98   |
| 10255   | 9          |            |             | 148.33  |
| 10256   | 3          |            |             | 13.97   |
| 10257   | 4          |            |             | 81.91   |
| 10258   | 1          | ERINSH     | Austria     | 140.51  |
| 10259   | 4          | CENTC      | Mexico      | 3.25    |

Sort Smallest to Largest

Sort Largest to Smallest

Clear Filter

Number Filters

Search

☒ (Select All)
 ☒ 1
 ☒ 2
 ☒ 3
 ☒ 4
 ☒ 5

OKCancel

1

2

3

4

5

6

7

8

...

1 of 17 pages (200 items)

#### Checkbox list generation:

By default, the checkbox list is generated from distinct values of the filter column from dataSource which gives an option to search and select the required items.

Also on checkbox list generation, if the number of distinct values are greater than 1000, then the excel filter will display only first 1000 values and show "Not all items shown" label to ensure the best performance on rendering and searching. However this limit has been customized according to your requirement by setting [filterSettings.maxFilterChoices](#) with required limit in integer.

**Note:** 1. Using excel filter events you can change the dataSource of the checkbox list.

2. [ej.Query](#) of checkbox list can also be changed using excel filter events.

The following code example describes the above behavior.

#### HTML

```
<div id="Grid"></div>
```

#### TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
  $(function () {
    var gridInstance = new ej.Grid($("#Grid"), {
      //The datasource "window.gridData" is referred from
      'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
    });
  });
}

```

```

dataSource: window["gridData"],
allowPaging: true,
allowSorting: true,
allowFiltering: true,
filterSettings: { filterType: "excel", maxFilterChoices: 4 },
columns: ["OrderID", "EmployeeID", "CustomerID", "ShipCountry", "Freight"]
});
});
}

```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID | CustomerID | ShipCountry | Freight |
|---------|------------|------------|-------------|---------|
| 10248   | 5          |            |             | 32.38   |
| 10249   | 6          |            |             | 11.61   |
| 10250   | 4          |            |             | 65.83   |
| 10251   | 3          |            |             | 41.34   |
| 10252   | 4          |            |             | 51.3    |
| 10253   | 3          |            |             | 58.17   |
| 10254   | 5          |            |             | 22.98   |
| 10255   | 9          |            |             | 148.33  |
| 10256   | 3          |            |             | 13.97   |
| 10257   | 4          |            |             | 81.91   |
| 10258   | 1          |            |             | 140.51  |
| 10259   | 4          | CENTC      | Mexico      | 3.25    |

Sort Smallest to Largest

Sort Largest to Smallest

Clear Filter

Number Filters

Search

Not all items showing

☒ (Select All)
☒ 1
☒ 2
☒ 3
☒ 4

OK Cancel

1

2

3

4

5

6

7

8

...

1 of 17 pages (200 items)

*Add current selection to filter checkbox:*

When filtering is done multiple times on the same column then the previously filtered values on the column will be cleared. So, to retain the old values **Add current selection to filter** checkbox can be used which is displayed when data is searched in the search bar.

The following image describes the above mentioned behavior.

| Order ID | Customer ID | Employee ID |
|----------|-------------|-------------|
| 10278    | BERGS       | 8           |
| 10280    | BERGS       | 2           |
| 10355    | AROUT       | 6           |
| 10365    | ANTON       | 3           |
| 10383    | AROUT       | 8           |
| 10384    | BERGS       | 3           |
| 10444    | BERGS       | 3           |
| 10445    | BERGS       | 3           |

Clear Filter  
Text Filters  
BLONP  
☒ (Select All)  
☒ Add current selection to filter  
☒ BLONP  
OK Cancel

1 of 1 pages (8 items)

### Case Sensitivity

To perform filter operation with case sensitive in excel styled filter menu mode by setting [enableCaseSensitivity](#) as **true**.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.gridData" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            allowPaging: true,
            allowFiltering: true,
            filterSettings: { enableCaseSensitivity: true, filterType: "excel" },
            columns: ["OrderID", "EmployeeID", "CustomerID", "ShipCountry", "Freight"]
        });
    });
}

```

The following output is displayed as a result of the above code example.

The screenshot shows a grid with columns: OrderID, EmployeeID, CustomerID, ShipCountry, and Freight. A 'Custom Filter' dialog box is open, showing a filter rule for 'CustomerID' with the expression 'Starts With' and the value 'france'. The dialog also has 'AND' and 'OR' radio buttons, and 'OK' and 'Cancel' buttons. The grid shows 17 pages (200 items) and a pagination bar at the bottom.

### Filter bar

[Filter bar](#) row is located next to column header of grid. You can filter the records with different expressions depending upon the column type. To show the filter bar, set the [filterType](#) as `filterbar`.

List of Filter bar Expressions:

You can enter the below filter expressions manually in the filter bar.

| Expression | Example  | Description  | Column Type |
|------------|----------|--|-------------|
| =          | = value  | equal  | Numeric     |
| !=         | != value | Not equal  |             |
| >          | > value  | Greater than   |             |
| <          | < value  | Less than  |             |
| >=         | >= value | Greater than or equal  | >           |
| <=         | <= value | Less than or equal   |             |
| N/A        | N/A      | Always <code>startswith</code> operator will be used for string filter | String      |
| N/A        | N/A      | Always <code>equal</code> operator will be used for Date filter        | Date        |
| N/A        | N/A      | Always <code>equal</code> operator will be used for Boolean filter     | Boolean     |



The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.gridData" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            allowPaging : true,
            allowFiltering : true,
            filterSettings : { filterType : "filterbar" },
            columns : ["OrderID", "EmployeeID", "CustomerID", "ShipCountry", "Freight"]
        });
    });
}
```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID | CustomerID | ShipCountry | Freight |
|---------|------------|------------|-------------|---------|
|         | >5         |            |             |         |
| 10249   | 6          | TOMSP      | Germany     | 11.61   |
| 10255   | 9          | RICSU      | Switzerland | 148.33  |
| 10262   | 8          | RATTC      | USA         | 48.29   |
| 10263   | 9          | ERNSH      | Austria     | 146.06  |
| 10264   | 6          | FOLKO      | Sweden      | 3.67    |
| 10268   | 8          | GROSR      | Venezuela   | 66.29   |
| 10271   | 6          | SPLIR      | USA         | 4.54    |
| 10272   | 6          | RATTC      | USA         | 98.03   |
| 10274   | 6          | VINET      | France      | 6.01    |
| 10276   | 8          | TORTU      | Mexico      | 13.84   |
| 10278   | 8          | BERGS      | Sweden      | 92.69   |
| 10279   | 8          | LEHMS      | Germany     | 25.83   |

⏪

⏩

1

2

3

4

5

6

⏭

⏮

⏯

1 of 6 pages (66 items)

EmployeeID: >5

Filter bar modes:

This specifies the grid to start the filter action while typing in the filter bar or after pressing the enter key based on [filterBarMode](#). There are two types of [filterBarMode](#), they are

1. OnEnter
2. Immediate

**Note:** For [filterBarMode](#) property you can assign either `string` value (onenter) or `enum` value (`ej.Grid.FilterBarMode.OnEnter`).

Filter bar message:

The filter bar message is supported only for the [filterType](#) as `filterbar`. The filtered data with column name is displayed in the grid pager itself. By default [showFilterBarStatus](#) is true.

The following code example describes the above behavior.

#### HTML

```
<div id="Grid"></div>
```

#### TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.gridData" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            allowPaging: true,
            allowFiltering: true,
            filterSettings: { showFilterBarStatus: false },
            columns: ["OrderID", "EmployeeID", "CustomerID", "ShipCountry", "Freight"
        ]});
    });
}

```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID | CustomerID | ShipCountry | Freight |
|---------|------------|------------|-------------|---------|
|         | 5          |            |             |         |
| 10248   | 5          | VINET      | France      | 32.38   |
| 10254   | 5          | CHOPS      | Switzerland | 22.98   |
| 10269   | 5          | WHITC      | USA         | 4.56    |
| 10297   | 5          | BLONP      | France      | 5.74    |
| 10320   | 5          | WARTH      | Finland     | 34.57   |
| 10333   | 5          | WARTH      | Finland     | 0.59    |
| 10358   | 5          | LAMAI      | France      | 19.64   |
| 10359   | 5          | SEVES      | UK          | 288.43  |
| 10372   | 5          | QUEEN      | Brazil      | 890.78  |
| 10378   | 5          | FOLKO      | Sweden      | 5.44    |
| 10397   | 5          | PRINI      | Portugal    | 60.26   |

1

1 of 1 pages (11 items)

EmployeeID: 5

## Filter Operators

The grid controls uses filter operators from [ej.DataManager](#), which are used at the time of filtering.

## List of Column type and Filter operators

| Column Type | Filter Operators |
|-------------|------------------|
|-------------|------------------|

|         |                                       |
|---------|---------------------------------------|
| Number  | ej.FilterOperators.greaterThan        |
|         | ej.FilterOperators.greaterThanOrEqual |
|         | ej.FilterOperators.lessThan           |
|         | ej.FilterOperators.lessThanOrEqual    |
|         | ej.FilterOperators.equal              |
|         | ej.FilterOperators.notEqual           |
| String  | ej.FilterOperators.startsWith         |
|         | ej.FilterOperators.endsWith           |
|         | ej.FilterOperators.contains           |
|         | ej.FilterOperators.equal              |
|         | ej.FilterOperators.notEqual           |
| Boolean | ej.FilterOperators.equal              |
|         | ej.FilterOperators.notEqual           |
| Date    | ej.FilterOperators.greaterThan        |
|         | ej.FilterOperators.greaterThanOrEqual |
|         | ej.FilterOperators.lessThan           |
|         | ej.FilterOperators.lessThanOrEqual    |
|         | ej.FilterOperators.equal              |
|         | ej.FilterOperators.notEqual           |

### FilterBar Template

Usually enabling allowFiltering, will create default textbox in Grid FilterBar. So, Using [filterBarTemplate](#) property of **columns** we can render any other controls like AutoComplete, DropDownList etc in filter bar to filter the grid data for the particular column.

It has three functions. They are

1. **create** - It is used to create the control at time of initialize.
2. **read** - It is used to read the Filter value selected.
3. **write** - It is used to render the control and assign the value selected for filtering.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

**TS**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: new ej.DataManager(window["gridData"]),
allowPaging: true,
allowFiltering: true,
columns: [
{ field: "OrderID", headerText: "Order ID", textAlign: ej.TextAlign.Right,
width: 75 },
{
field: "CustomerID", headerText: "CustomerID", width: 100,
filterBarTemplate: {
create: function (args) {
return "<input>"
},
write: function (args) {
var data = this.model.dataSource.executeLocal(new
ej.Query().select("CustomerID"));
args.element.ejAutocomplete({ width: "100%", dataSource: data,
enableDistinct: true, focusOut: ej.proxy(args.column.filterBarTemplate.read,
this, args) });
},
read: function (args) {
this.filterColumn(args.column.field, "equal", args.element.val(), "and",
true)
},
},
},
{
field: "EmployeeID", headerText: "EmployeeID", width: 100,
filterBarTemplate: {
write: function (args) {
var data = [{ text: "clear", value: "clear" }, { text: "1", value: 1 }, {
text: "2", value: 2 }, { text: "3", value: 3 }, { text: "4", value: 4 },
{ text: "5", value: 5 }, { text: "6", value: 6 }, { text: "7", value: 7 }, {
text: "8", value: 8 }, { text: "9", value: 9 }
]
args.element.ejDropDownList({ width: "100%", dataSource: data, change:
ej.proxy(args.column.filterBarTemplate.read, this, args) })
},
read: function (args) {
if (args.element.val() == "clear") {
this.clearFiltering(args.column.field);
args.element.val("")
}
this.filterColumn(args.column.field, "equal", args.element.val(), "and",
true)
},
},
},
{

```

```

field: "Freight", headerText: "Freight", width: 100, filterBarTemplate: {
write: function (args) {
args.element.ejNumericTextbox({ width: "100%", decimalPlaces: 2, focusOut:
ej.proxy(args.column.filterBarTemplate.read, this, args) });
},
read: function (args) {
this.filterColumn(args.column.field, "equal", args.element.val(), "and",
true)
},
},
},
{ field: "ShipCountry", headerText: "Ship Country", width: 90 },
{
field: "Verified", textAlign: ej.TextAlign.Center, headerText: "Verified",
width: 80, filterBarTemplate: {
write: function (args) {
args.element.ejCheckBox({ change:
ej.proxy(args.column.filterBarTemplate.read, this, args) });
},
read: function (args) {
this.filterColumn(args.column.field, "equal",
args.element.parent().attr('aria-checked'), "and", true)
},
},
}
}
});
});
}

```

The following output is displayed as a result of the above code example.

| Order ID             | CustomerID           | EmployeeID                             | Freight                                | Ship Country         | Verified                            |
|----------------------|----------------------|--|--|----------------------|-------------------------------------|
| <input type="text"/> | <input type="text"/> | clear <input type="button" value="v"/> | 58.17 <input type="button" value="v"/> | <input type="text"/> | <input type="checkbox"/>            |
| 10253                | HANAR                | 3                                      | 58.17                                  | Brazil               | <input checked="" type="checkbox"/> |

1 of 1 pages (1 items)

Freight: 58.17

After Filtering

### Selection

The Selection provides an interactive support to highlight the row, cell or column that you select. Selection can be done through simple Mouse down or Keyboard interaction. To enable selection, set the [allowSelection](#) as **true**.

### Types of Selection

There are two types of selections available in Grid. They are as follows:

1. Single
2. Multiple

### Single Selection

Single selection is an interactive support to select a specific row, cell or column in grid by mouse or keyboard interactions. To enable single selection set the [selectionType](#) property as `single` and also set the [allowSelection](#) property as `true`.

### Multiple Selections

Multiple selections is an interactive support to select a group of rows, cells or columns in grid by mouse or keyboard interactions. To enable multiple selections set the [selectionType](#) property as `multiple` and also set the [allowSelection](#) property as `true`.

### Row Selection

Row selection is enabled by setting the [selectionMode](#) property of [selectionSettings](#) as `row`. For random row selection, press **"Ctrl + mouse left"** click and for continuous row selection press **"Shift + mouse left"** click on the grid rows. To unselect the selected rows, press **"Ctrl + mouse left"** click on the selected row.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.gridData" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            allowPaging : true,
            allowSelection : true,
            selectionType : "multiple",
            selectionSettings: {selectionMode: ["row"] },
            columns : ["OrderID", "EmployeeID", "ShipCity", "ShipCountry", "Freight"]
        });
    });
}
```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID | ShipCity       | ShipCountry | Freight |
|---------|------------|----------------|-------------|---------|
| 10248   | 5          | Reims          | France      | 32.38   |
| 10249   | 6          | Münster        | Germany     | 11.61   |
| 10250   | 4          | Rio de Janeiro | Brazil      | 65.83   |
| 10251   | 3          | Lyon           | France      | 41.34   |
| 10252   | 4          | Charleroi      | Belgium     | 51.3    |
| 10253   | 3          | Rio de Janeiro | Brazil      | 58.17   |
| 10254   | 5          | Bern           | Switzerland | 22.98   |
| 10255   | 9          | Genève         | Switzerland | 148.33  |
| 10256   | 3          | Resende        | Brazil      | 13.97   |
| 10257   | 4          | San Cristóbal  | Venezuela   | 81.91   |
| 10258   | 1          | Graz           | Austria     | 140.51  |
| 10259   | 4          | México D.F.    | Mexico      | 3.25    |

1

2

3

4

5

6

7

8

...

1

2

1 of 17 pages (200 items)

### Multiple Row Selection using Checkbox Column

Select multiple rows in grid by using the Checkbox column and it can be enabled by setting column **type** as **checkbox**. It also provides the option to select/deselect all the rows in Grid using a checkbox in the corresponding column header.

If the **field** property of Checkbox column is not defined, then it acts as a template column. So, the **field** property is necessary to perform grid actions like sorting, editing, etc., for the corresponding Checkbox column.

The following code example describes the above behavior.

#### HTML

```
<div id="Grid"></div>
```

#### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
allowPaging: true,
columns: [
```



```

{ type: "checkbox", width: 50 }, //enables the checkbox column
{ field: "OrderID", isPrimaryKey: true, width: 80, textAlign:
ej.TextAlign.Right },
{ field: "CustomerID", headerText: "Customer ID", width: 75 },
{ field: "EmployeeID", headerText: "Employee ID", width: 75, textAlign:
ej.TextAlign.Right },
{ field: "Freight", headerText: "Freight", width: 75, textAlign:
ej.TextAlign.Right, format: "{0:C}" },
]
});
});
}

```

The following output is displayed as a result of the above code example.

|                                     | OrderID | Customer ID | Employee ID | Freight  |
|-------------------------------------|---------|-------------|-------------|----------|
| <input type="checkbox"/>            | 10248   | vINET       | 5           | \$32.38  |
| <input checked="" type="checkbox"/> | 10249   | tOMSP       | 6           | \$11.61  |
| <input type="checkbox"/>            | 10250   | hANAR       | 4           | \$65.83  |
| <input checked="" type="checkbox"/> | 10251   | vICTE       | 3           | \$41.34  |
| <input checked="" type="checkbox"/> | 10252   | sUPRD       | 4           | \$51.30  |
| <input type="checkbox"/>            | 10253   | hANAR       | 3           | \$58.17  |
| <input type="checkbox"/>            | 10254   | CHOPS       | 5           | \$22.98  |
| <input type="checkbox"/>            | 10255   | RICSU       | 9           | \$148.33 |
| <input type="checkbox"/>            | 10256   | WELLI       | 3           | \$13.97  |
| <input type="checkbox"/>            | 10257   | HILAA       | 4           | \$81.91  |
| <input type="checkbox"/>            | 10258   | ERNSH       | 1           | \$140.51 |
| <input type="checkbox"/>            | 10259   | CENTC       | 4           | \$3.25   |

1 of 17 pages (200 items)

### Cell Selection

Cell selection is enabled by setting the [selectionMode](#) property of [selectionSettings](#) as `cell`. For random cell selection, press **“Ctrl + mouse left”** click and for continuous cell selection, press **“Shift + mouse left”** click on the grid cells. To unselect selected cells, press **“Ctrl + mouse left”** on the selected cell.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
allowPaging: true,
allowSelection: true,

```

```

selectionType: "multiple",
selectionSettings: { selectionMode: ["cell"] },
columns: ["OrderID", "EmployeeID", "ShipCity", "ShipCountry", "Freight"]
});
});
}

```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID | ShipCity       | ShipCountry | Freight |
|---------|------------|----------------|-------------|---------|
| 10248   | 5          | Reims          | France      | 32.38   |
| 10249   | 6          | Münster        | Germany     | 11.61   |
| 10250   | 4          | Rio de Janeiro | Brazil      | 65.83   |
| 10251   | 3          | Lyon           | France      | 41.34   |
| 10252   | 4          | Charleroi      | Belgium     | 51.3    |
| 10253   | 3          | Rio de Janeiro | Brazil      | 58.17   |
| 10254   | 5          | Bern           | Switzerland | 22.98   |
| 10255   | 9          | Genève         | Switzerland | 148.33  |
| 10256   | 3          | Resende        | Brazil      | 13.97   |
| 10257   | 4          | San Cristóbal  | Venezuela   | 81.91   |
| 10258   | 1          | Graz           | Austria     | 140.51  |
| 10259   | 4          | México D.F.    | Mexico      | 3.25    |

1
2
3
4
5
6
7
8
...

1 of 17 pages (200 items)

#### Cell Selection Mode

There are two types of cell selection available in grid. They are as follows:

1. Continuous Selection
2. Box Selection

Box cell selection is to select multiple cells vertically based on the initial column index selection.

The following code example describes the above behavior.

#### HTML

```
<div id="Grid"></div>
```

#### TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />

```

```

module GridComponent {
  $(function () {
    var gridInstance = new ej.Grid($("#Grid"), {
      //The datasource "window.gridData" is referred from
      'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
      dataSource: window["gridData"],
      allowPaging: true,
      allowSelection: true,
      selectionType: "multiple",
      selectionSettings: { selectionMode: ["cell"], cellSelectionMode:
        ej.Grid.CellSelectionMode.Box },
      columns: ["OrderID", "EmployeeID", "ShipCity", "ShipCountry", "Freight"]
    });
  });
}

```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID | ShipCity       | ShipCountry | Freight |
|---------|------------|----------------|-------------|---------|
| 10248   | 5          | Reims          | France      | 32.38   |
| 10249   | 6          | Münster        | Germany     | 11.61   |
| 10250   | 4          | Rio de Janeiro | Brazil      | 65.83   |
| 10251   | 3          | Lyon           | France      | 41.34   |
| 10252   | 4          | Charleroi      | Belgium     | 51.3    |
| 10253   | 3          | Rio de Janeiro | Brazil      | 58.17   |
| 10254   | 5          | Bern           | Switzerland | 22.98   |
| 10255   | 9          | Genève         | Switzerland | 148.33  |
| 10256   | 3          | Resende        | Brazil      | 13.97   |
| 10257   | 4          | San Cristóbal  | Venezuela   | 81.91   |
| 10258   | 1          | Graz           | Austria     | 140.51  |
| 10259   | 4          | México D.F.    | Mexico      | 3.25    |

1
2
3
4
5
6
7
8
...

1 of 17 pages (200 items)

### Column Selection

[Column selection](#) can be enabled by setting the [selectionMode](#) property of [selectionSettings](#) as `column`. For random column selection, press **“Ctrl + mouse left click”** and for continuous column selection, press **“Shift + mouse left click”** on the top of the column header. To unselect selected columns, press **“Ctrl + mouse left click”** on top of the selected column header.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

## TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
$(function () {
var gridInstance = new ej.Grid($("#Grid"), {
//The datasource "window.gridData" is referred from
'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
dataSource: window["gridData"],
allowPaging: true,
allowSelection: true,
selectionType: "multiple",
selectionSettings: { selectionMode: ["column"] },
columns: ["OrderID", "EmployeeID", "ShipCity", "ShipCountry", "Freight"]
});
});
}
}

```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID | ShipCity       | ShipCountry | Freight |
|---------|------------|----------------|-------------|---------|
| 10248   | 5          | Reims          | France      | 32.38   |
| 10249   | 6          | Münster        | Germany     | 11.61   |
| 10250   | 4          | Rio de Janeiro | Brazil      | 65.83   |
| 10251   | 3          | Lyon           | France      | 41.34   |
| 10252   | 4          | Charleroi      | Belgium     | 51.3    |
| 10253   | 3          | Rio de Janeiro | Brazil      | 58.17   |
| 10254   | 5          | Bern           | Switzerland | 22.98   |
| 10255   | 9          | Genève         | Switzerland | 148.33  |
| 10256   | 3          | Resende        | Brazil      | 13.97   |
| 10257   | 4          | San Cristóbal  | Venezuela   | 81.91   |
| 10258   | 1          | Graz           | Austria     | 140.51  |
| 10259   | 4          | México D.F.    | Mexico      | 3.25    |

1
2
3
4
5
6
7
8
...

1 of 17 pages (200 items)

### Touch options

While using the Grid in a [touch](#) device environment, there is an option for multi selection through single tap on the row and it will show a popup with multi-selection symbol. Tap the icon to enable multi selection in a single tap.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.gridData" is referred from
            'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            allowPaging : true,
            allowSelection : true,
            selectionType : "multiple",
            columns : ["OrderID", "EmployeeID", "ShipCity", "ShipCountry", "Freight"]
        });
    });
}
```

The following output is displayed as a result of the above code example.

| OrderID | EmployeeID | ShipCity       | ShipCountry | Freight |
|---------|------------|----------------|-------------|---------|
| 10248   | 5          | Reims          | France      | 32.38   |
| 10249   | 6          | Münster        | Germany     | 11.61   |
| 10250   | 4          | Rio de Janeiro | Brazil      | 65.83   |
| 10251   | 3          | Lyon           | France      | 41.34   |
| 10252   | 4          | Charleroi      | Belgium     | 51.3    |
| 10253   | 3          | Rio de Janeiro | Brazil      | 58.17   |
| 10254   | 5          | Bern           | Switzerland | 22.98   |
| 10255   | 9          | Genève         | Switzerland | 33      |
| 10256   | 3          | Resende        | Brazil      | 13.97   |
| 10257   | 4          | San Cristóbal  | Venezuela   | 81.91   |
| 10258   | 1          | Graz           | Austria     | 140.51  |
| 10259   | 4          | México D.F.    | Mexico      | 3.25    |

1
2
3
4
5
6
7
8
...

1 of 17 pages (200 items)

### Toggle Selection

The [Toggle](#) selection allows to perform selection and unselection of the particular row, cell or column. To enable toggle selection, set [enableToggle](#) property of the [selectionSettings](#) as `true`. If you click on the selected row, cell or column then it will be unselected and vice versa.

**Note:** If multi selection is enabled, then first click on any selected row (without pressing Ctrl key), it will clear the multi selection and in second click on the same row, it will be unselected.

The following code example describes the above behavior.

### HTML

```
<div id="Grid"></div>
```

### TS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GridComponent {
    $(function () {
        var gridInstance = new ej.Grid($("#Grid"), {
            //The datasource "window.gridData" is referred from
            // 'http://js.syncfusion.com/demos/web/scripts/jsondata.min.js'
            dataSource: window["gridData"],
            allowPaging: true,
            enableRowHover: false,
            selectionSettings: { enableToggle: true },
            columns: ["OrderID", "EmployeeID", "ShipCity", "ShipCountry", "Freight"]
        });
    });
}

```

```
});  
});  
}
```

## GroupBox

### Overview

GroupBox is a UI component which provides the option to have the multiple buttons in succession. This component also has data source binding and more flexible options to provide the friendlier implementation and usage. Different modes with GroupButton gives the flexible and rich options to manage the actions based on the application needs.

### Key Features

- Trendy Look: Rich Appearance with Theme Support
- RTL: Supports for Right to Left alignment
- Different Modes of Button: Supports the rising of click event Repeatedly
- DataSource: Supports Data binding with JSON data and remote data.
- Easy Customization: The customization of Button control to any form is made simple

### Creating a GroupButton in TypeScript

You can create a **TypeScript** application with the help of the given

<https://help.syncfusion.com/js/typescript>. In application, Within an index.html file add the scripts references in the order mentioned in the above link.

This can be created from an input element with the HTML id attribute and pre-defined options set to it.

### HTML

```
<div class="row">  
  <div class="cols-sample-area">  
    <div class="control">  
      <label>Appointment View</label>  
      <div class="element">  
        <div id="GroupBox"></div>  
      </div>  
    </div>  
  </div>  
</div>
```

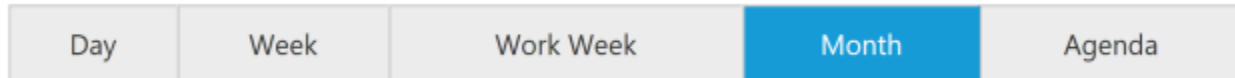
Create app.ts file and past the below content

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module GroupButtonComponent {  
  $(function () {  
    new ej.GroupButton($("#GroupBox"), {  
      dataSource: [  
        { text: "Day", contentType: "textonly" },  
        { text: "Week", contentType: "textonly" },  
        { text: "Work Week", contentType: "textonly" },  
      ],  
    });  
  });  
}
```

```
{ text: "Month", contentType: "textonly", selected: "selected" },
{ text: "Agenda", contentType: "textonly" }]
};
});
}
```

Now build your application, so that the **app.js** is automatically generated and got added to your project (User have nothing to do with this file). Now, whatever code changes that you make in **app.ts** file will be reflected in **app.js** file automatically.



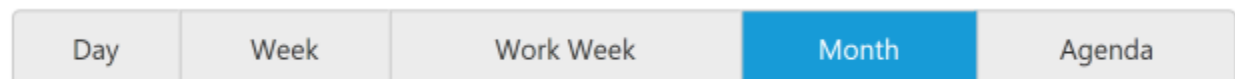
### Configuring APIs in GroupButton

All GroupButton properties, can be configured and used in TypeScript platform, with types specification. Please refer the below code example to know the configuring the APIs of GroupButton in typescript platform

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
$(function () {
new ej.GroupButton($("#GroupButton"), {
dataSource: [
{ text: "Day", contentType: "textonly" },
{ text: "Week", contentType: "textonly" },
{ text: "Work Week", contentType: "textonly" },
{ text: "Month", contentType: "textonly", selected: "selected" },
{ text: "Agenda", contentType: "textonly" }],
showRoundedCorner: true,
selectedIndex: [4]
}
);
});
}
```

The following code will render the following output.



### Behavior Settings

GroupButton has some default behavior settings which helps you to perform more operation by Built-in selection.

#### Different modes of button

GroupButton provides the different types of modes called check box mode, radio button mode. Setting a **groupButtonMode** property value to checkbox mode, we can perform the multiple actions in a single



group. In this case, you can toggle the all the buttons state and perform the actions, since it all will behave as individual button in a group of button.

Setting a ButtonMode to radio, we can perform the only single action with all related actions in a group of button.

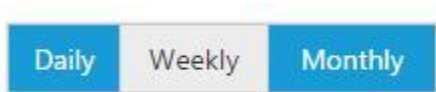
**Checkbox Mode :** Here, multiple button can be selected, please check with the below use case for this mode.

### HTML

```
<div id="groupButton">
<ul>
<li>
Daily
</li>
<li>
Weekly
</li>
<li>
Monthly
</li>
</ul>
</div>
```

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
$(function () {
var groupButton = new ej.GroupButton($("#groupButton"), {
groupButtonMode: "checkbox",
width: "215px"
});
});
}
```



**Radio Button Mode:** Here, single button only can be selected, Please check with the below use case

### HTML

```
<div id="groupButton">
<ul>
<li>
Credit Card
</li>
<li>
Debit Card
</li>
<li>
Net Banking
</li>
</ul>
</div>
```

```

</li>
</ul>
</div>

```

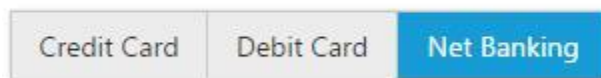
## JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
  $(function () {
    var groupButton = new ej.GroupButton($("#groupButton"), {
      groupButtonMode: "radiobutton",
      width: "215px"
    });
  });
}

```

Choose payment mode



## SelectedItemIndex

The state of the button can be changed by clicking on the button. Also the selection state of the button can be achieved by using **selectedItemIndex** API which is used to select the button items in GroupButton based on index.

This property will accept the array values and its value will be differ based on current mode of Button. If the Button mode is "Radio", then the selected items can have single value within the array, since we can select a single button only in this mode. Whereas for the "checkbox" mode, the selectedItemIndex can have the multiple values within an array, since multiple button can be in active state in this mode.

### Setting a selectedItemIndex for checkbox mode

## HTML

```

<div id="groupButton">
  <ul>
    <li>
      Credit Card
    </li>
    <li>
      Debit Card
    </li>
    <li>
      Net Banking
    </li>
  </ul>
</div>

```

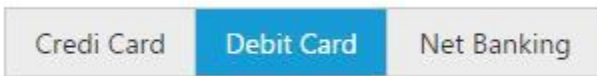
### Setting a selectedItemIndex for radio button mode

## JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
    $(function () {
        var groupButton = new ej.GroupButton($("#groupButton"), {
            groupButtonMode: "radiobutton",
            width: "215px",
            selectedItemIndex: [1]
        });
    });
}

```



Select and deselect the button items using public method

Button items in GroupButton can be selected or deselected using available public methods “**selectItem**”, **deselectItem**”. These two methods will get the index of targeted button as argument.

### Select the Button Items

## HTML

```

<div id="groupButton">
<ul>
<li>
Credit Card
</li>
<li>
Debit Card
</li>
<li>
Net Banking
</li>
</ul>
</div>

```

## JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
    $(function () {
        var groupButton = new ej.GroupButton($("#groupButton"), {
            groupButtonMode: "radiobutton",
            width: "300px",
            selectedItemIndex: [1]
        });
        groupButton.selectItem(0);
    });
}

```

## Deselect the Button Items

### JS

```
groupButton.deselectItem(0);
```

## Select or deselect using id attribute

You can select or deselect the button items based on id of the button items also. Please check with the below code example to know about the selection/deselection based on id attribute

### HTML

```
<div id="groupButton">
<ul>
<li id="payment_option_1">
Credit Card
</li>
<li id="payment_option_2">
Debit Card
</li>
<li id="payment_option_3">
Net Banking
</li>
</ul>
</div>
```

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
$(function () {
var groupButton = new ej.GroupButton($("#groupButton"), {
groupButtonMode: "checkbox",
width: "300px",
});
groupButton.selectItem("payment_option_2");
});
}
```

## Customization

### Size

GroupButton component size can be varied based on the Size property values. Size is the enum type API and it has the following Built-in values

List of predefined button size

| Button Types | Description  |
|--------------|--|
| Normal       | Creates GroupButton with content size.                               |
| Mini         | Creates GroupButton with Built-in mini size height, width specified. |

|        |  |
|--------|--|
| Small  | Creates GroupButton with Built-in small size height, width specified.  |
| Medium | Creates GroupButton with Built-in medium size height, width specified. |
| Large  | Creates GroupButton with Built-in large size height, width specified.  |

### Set Dimension

By default GroupButton has standard height and width. You can change this height and width by using height and width property respectively.

### JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
  $(function () {
    var groupButton = new ej.GroupButton($("#groupButton"), {
      groupButtonMode: "checkbox",
      width: "300px",
      height: "50px",
      selectedItemIndex: [0]
    });
  });
}

```



### ContentType

The content of the **Button** items in GroupButton can have a text and images. GroupButton provides the some predefined contentType options to easily customize the appearance of each button associated with GroupButton component without any complex CSS tricks. **GroupButton** items supports the following content types.

List of content types for button

| Content Types  | Description                                       |
|----------------|---|
| textOnly       | Supports only for text content only.              |
| imageOnly      | Supports only for image content only              |
| imageBoth      | Supports image for both ends of the button.       |
| textAndImage   | Supports image with the text content.             |
| imageTextImage | Supports image with both ends and middle in text. |

## Icons

GroupButton has the option to add the icons to button elements which enhance the appearance. Icons inside the button can be added easily using **prefixIcon** and **suffixIcon** fields with **dataSource** property. GroupButton control also supports the Built-in icon libraries. The `ej.widgets.core.min.css` contains definitions for important icons that can be used in buttons. You can get the details about available icons with that corresponding class from [here](#).

Simply you can use these Built-in icons by mentioning the icon class name as value in **prefixIcon** and **suffixIcon** property. You can use any font icons that are defined in `ej.widgets.core.min.css`. It avoids the complexity in specifying icon using sprite image and CSS.

## HTML

```
<div class="element">
  <div id="groupButton">
  </div>
</div>
```

## JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
  $(function () {
    var groupButton = new ej.GroupButton($("#groupButton"), {
      groupButtonMode: "checkbox",
      dataSource: [
        { contentType: "imageonly", prefixIcon: "e-icon e-bold_01" },
        { contentType: "imageonly", prefixIcon: "e-icon e-italic_01" },
        { contentType: "imageonly", prefixIcon: "e-icon e-underline_01" } ],
      showRoundedCorner: true,
      width: "170px",
      selectedItemIndex: [0, 2]
    });
  });
}
```

### Apply Styles



## Orientation

GroupButton has two Built-in orientation support called vertical and horizontal orientations which defines the direction of rendered GroupButton component. You can set the value to this property as enum or string type.

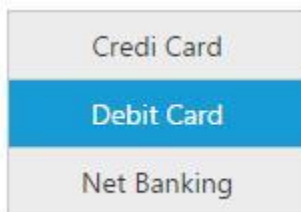
- `ej.Orientation.Horizontal` or "Horizontal"
- `ej.Orientation.Vertical` or "Vertical"

## JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
    $(function () {
        var groupButton = new ej.GroupButton($("#groupButton"), {
            groupButtonMode: "checkbox",
            orientation: ej.Orientation.Vertical,
            selectedIndex: [1]
        });
    });
}

```



## DataSource

GroupButton can populate the button items based on data source and by specifying the associated fields.

Refer the below table to know about the available fields

|               |  |
|---------------|--|
| text          | Text to be displayed in button   |
| prefixIcon    | Icon class name “ prefixIcon will be displayed from the left margin of the button. |
| suffixIcon    | Icon class name “ suffixIcon will be displayed from the left margin of the button. |
| contentType   | Specifies content type of button item  |
| imagePosition | Specifies position of the image in a button item                                   |
| Selected      | Specifies the selection state of button item                                       |
| URL           | Used to include the URL tag to the button item                                     |
| htmlAttribute | It defines the HTML attributes such as class and styles for an button item.        |
| linkAttribute | It defines the image attributes such as height, width, styles, etc.                |

## Local Data

To set the local JSON data, define a JSON array and initialize the GroupButton with **dataSource** property. Specify the column names in the fields’ property.

**Note:** the columns are bounded automatically when the fields are specified with the default names like id, text, etc...

Below is the sample to code to render the GroupButton JSON dataSource,

### HTML

```
<div id="groupButton">
</div>
```

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
$(function () {
var groupButton = new ej.GroupButton($("#groupButton"), {
groupButtonMode: "radiobutton",
dataSource: [
{ text: "Day", contentType: "textonly" },
{ text: "Week", contentType: "textonly" },
{ text: "Work Week", contentType: "textonly" },
{ text: "Month", contentType: "textonly", selected: "selected" },
{ text: "Agenda", contentType: "textonly" }],
showRoundedCorner: true
});
});
}
</script>
```

### Appointment View



### Remote Data

To bind remote data to the GroupButton, you can assign a service data as an instance of `ejDataManager` to the `dataSource` property along with the fields mapping.

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
var dataManger = ej.DataManager({
url: "http://mvc.syncfusion.com/Services/Northwnd.svc/"
});
// Query creation
var query = ej.Query().from("Orders").take(6);
$(function () {
// declaration
var groupButton = new ej.GroupButton($("#groupButton"), {
dataSource: dataManger,
fields: { text: "CustomerID" },
});
});
}
```



```
query: query,
});
});
}
</script>
```

## Miscellaneous

### Show/Hide the items

Particular currently showing button items can be hidden. Also it provides the options to show the hidden button again. These functionalities can be achieved using **showItem** or **hideItem** method.

#### Hide the Button item based on given index

##### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
$(function () {
var groupButton = new ej.GroupButton($("#groupButton"), {
});
groupButton.hideItem(1);
});
}
```

#### Show the hidden Button item based on given index

##### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
$(function () {
var groupButton = new ej.GroupButton($("#groupButton"), {
});
groupButton.showItem(1);
});
}
```

Also entire group button, can be hide/show using public methods **hide()**, **show()**.

### Enable/Disable

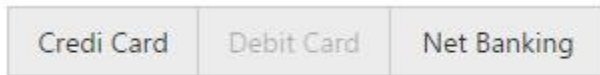
Particular Items can be enabled/disabled using **enableItem**, **disableItem** methods. This takes the index of the button as the argument.

Also entire GroupButton can be enabled or disabled using **enable ()**, **disable** public method.

##### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
$(function () {
var groupButton = new ej.GroupButton($("#groupButton"), {
});
});
}
```

```
groupButton.disableItem(1);
});
}
```



### Getting Index of given Element

By passing the jQuery element of the required button to **getIndex** public method, we can get the index of that passed button element.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
$(function () {
var groupButton = new ej.GroupButton($("#groupButton"), {
});
groupButton.getIndex(id);
});
}
```

### Getting state of given Button

You can get the selection state of required button by passing that button jQuery element to **isSelected** public method.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
$(function () {
var groupButton = new ej.GroupButton($("#groupButton"), {
});
groupButton.selectItem(1);
alert(groupButton.isSelected(1));
});
}
```

Also you can get the active / disabled state required button by passing that button jQuery element to **isDisabled** public method.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module GroupButtonComponent {
$(function () {
var groupButton = new ej.GroupButton($("#groupButton"), {
});
groupButton.disableItem(1);
alert(groupButton.isDisabled(1));
});
}
```

}

## HeatMap

### Overview

**Essential HeatMap TypeScript** represents tabular data values as gradient colors instead of numbers. Low and high values are different colors with different gradients.

| Product Name         | Y2010 | Y2011 | Y2012 | Y2013 | Y2014 | Y2015 | Y2016 | Y2017 | Y2018 |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Vegie-spread         | 24    | 27    | 89    | 53    | 29    | 97    | 18    | 22    | 9     |
| Tofuua               | 25    | 82    | 7     | 32    | 36    | 84    | 2     | 51    | 5     |
| Alice Mutton         | 44    | 13    | 43    | 13    | 60    | 21    | 71    | 17    | 1     |
| Konbu                | 12    | 45    | 66    | 7     | 99    | 73    | 49    | 14    | 7     |
| Fløtemysost          | 66    | 1     | 74    | 42    | 8     | 1     | 40    | 6     | 3     |
| Perth Pasties        | 90    | 50    | 89    | 36    | 62    | 97    | 91    | 96    | 2     |
| Boston Crab Meat     | 92    | 39    | 17    | 70    | 17    | 52    | 88    | 24    | 6     |
| Raclette Courdavault | 70    | 65    | 99    | 58    | 96    | 8     | 8     | 63    | 1     |

Key features:

- 2 types of data source mapping (TableMapping, CellMapping)
- Color mapping
- Legend
- Virtualization

### Getting Started

This section helps to get started of the HeatMap component for TypeScript.

#### Initialize HeatMap

The HeatMap can be created from a HTML 'div' element. To create the HeatMap, you should call the 'ejHeatMap' jQuery plug-in function.

#### HTML

```
<!DOCTYPE html>
<html>
<body>
<script type="text/javascript" src="heatmap/heatmap.js">
</script>
<div id="heatmap"></div>
</body>
</html>
```

#### Prepare and Populate data

Populate product information in a collection called **ItemsSource**.

### Map data into HeatMap

Now data is ready, next we need to configure data source and map rows and columns to visualize. For that, need to prepare `ItemsMapping` add it in resource and set items source and mapping.

Next we can configure color range for these values using color mapping and also configure items mapping based on items source.

### JAVASCRIPT

```
$(function () {
    var heatmap = new ej.datavisualization.HeatMap($("#heatmap"), {
        colorMappingCollection: [
            { value: 0, color: "#8ec8f8", label: { text: "0" } },
            { value: 100, color: "#0d47a1", label: { text: "100" } }
        ],
        isResponsive: true,
        itemsSource: itemSource,
        width: "100%",
        itemsMapping: {
            column: { propertyName: "ProductName", displayName: "Product Name" },
            row: { propertyName: "Year", displayName: "Year" },
            value: { propertyName: "Value" },
            columnMapping: [
                { "propertyName": columns[0], "displayName": columns[0] },
                { "propertyName": columns[1], "displayName": columns[1] },
                { "propertyName": columns[2], "displayName": columns[2] },
                { "propertyName": columns[3], "displayName": columns[3] },
                { "propertyName": columns[4], "displayName": columns[4] },
                { "propertyName": columns[5], "displayName": columns[5] }
            ],
            headerMapping: { propertyName: "Year", displayName: "Year", columnStyle: {
                width: 105, textAlign: "right" } },
        },
    });
});
```

| Year  | Veggie-spread | Tofuua | Alice Mutton | Konbu | Fløtemysost |
|-------|---------------|--------|--------------|-------|-------------|
| Y2011 | 36            | 38     | 2            | 60    | 44          |
| Y2012 | 64            | 32     | 95           | 10    | 76          |
| Y2013 | 62            | 25     | 75           | 1     | 8           |
| Y2014 | 6             | 15     | 76           | 81    | 14          |
| Y2015 | 80            | 80     | 39           | 52    | 42          |
| Y2016 | 97            | 74     | 94           | 27    | 87          |

### Initialize Legend

A legend control is used to represent range value in a gradient, create a legend with the same color mapping as shown below.

### HTML

```
<!DOCTYPE html>
<html>
<body>
<script type="text/javascript" src="heatmap/heatmap.js">
</script>
```

```
<div id="heatmap_legend"></div>
</body>
</html>
```

## JAVASCRIPT

```
$(function () {
  var heatmaplegend = new
  ej.datavisualization.HeatMapLegend($("#heatmap_legend"), {
    colorMappingCollection: [
      { value: 0, color: "#8ec8f8", label: { text: "0" } },
      { value: 100, color: "#0d47a1", label: { text: "100" } }
    ],
    height: "50px",
    width: "75%",
    isResponsive: true
  });
});
```



## Kanban Board

### Overview

The Kanban control for JavaScript is an efficient way to visualize the workflow at each stage along its path to completion. The most important features available are Swim lane, filtering, and editing.

Some important features of the Kanban control are:

- \* *Data sources* - Bind the Kanban control with an array of JSON objects or [ej.DataManager](#) which support OData and remote web service binding.
- \* *Swim lane* – Supports Swim lane grouping. *Filters* – Supports filtering based on user Query. *Editing* - Offers card editing, inserting, and deleting using Dialog.
- \* *Card template* - Offers to render the card based on user template.

### Getting Started

For common getting started of typescript , you can refer [here](#).

The default type definition file ej.web.all.d.ts needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the intelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the jquery.d.ts file in your TypeScript application, which can be downloaded from [here](#).

### Preparing HTML document

The Kanban control has the following list of external JavaScript dependencies.

- [jQuery 1.7.1](#) and later versions
- [jsRender](#) - to render the templates

### Adding Script Reference

Refer to the internal dependencies in the following table.

| Files                    | Description/Usage  |
|--------------------------|--|
| ej.core.min.js           | It is referred always before using all the JS controls.                          |
| ej.data.min.js           | Used to handle data operation and is used while binding data to the JS controls. |
| ej.touch.min.js          | It is referred when using touch functionalities in Kanban.                       |
| ej.draggable.min.js      | It is referred when using drag and drop in Kanban.                               |
| ej.kanban.min.js         | The Kanban™'s main file.   |
| ej.globalize.min.js      | It is referred when using localization in Kanban.                                |
| ej.scroller.min.js       | It is referred when scrolling is used in the Kanban.                             |
| ej.waitingpopup.min.js   | It is referred when waiting popup used.  |
| ej.dropdownlist.min.js   | These files are used while enable the Editing feature in the Kanban.             |
| ej.dialog.min.js         |  |
| ej.button.min.js         |  |
| ej.datepicker.min.js     |  |
| ej.datetimepicker.min.js |  |
| ej.editor.min.js         |  |
| ej.toolbar.min.js        | These files are used while enable the Filtering feature in the Kanban.           |
| ej.menu.min.js           | These files are used while enable the context menu feature in the Kanban.        |
| ej.checkbox.min.js       |  |
| ej.rte.min.js            | These files are used while using the cell edit type as RTE in the Kanban.        |

To get started, you can use the `ej.web.all.min.js` file that encapsulates all the `ej` controls and frameworks in one single file. So the complete boilerplate code is

### HTML

```
<!DOCTYPE html>
<html>
```

```

<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential Studio for JavaScript">
<meta name="author" content="Syncfusion">
<title></title>
<!-- Essential Studio for JavaScript theme reference -->
<link href="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<!-- Essential Studio for JavaScript script references -->
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jquery.globalize.min.js">
</script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"></script>
<!-- Add your custom scripts here -->
</head>
<body>
</body>
</html>

```

### Create a Kanban

The Kanban can be created from a HTML DIV element with the HTML id attribute set to it. To create the Kanban, you should call the `ejKanban` jQuery plug-in function with the options as parameter. Refer to the following code example.

#### HTML

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var sample = new ej.Kanban($("#Kanban"), {
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
]
});
});
}

```

Backlog

In Progress

Done

No cards to display

### Data Binding

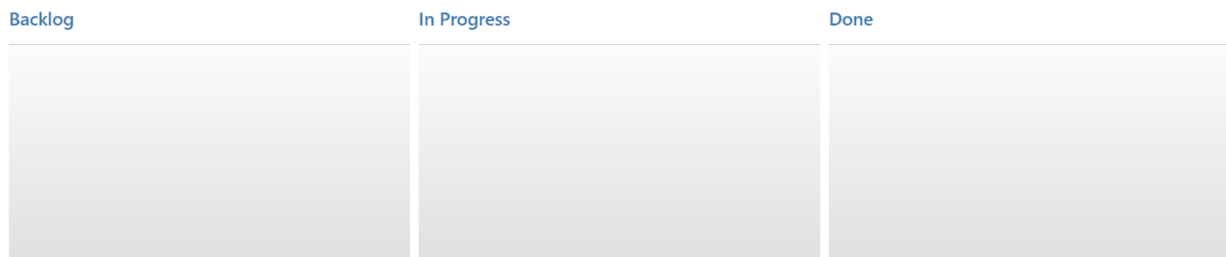
Data binding in the Kanban is achieved by using the `ej.DataManager` that supports both RESTful JSON data services binding and local JSON array binding. To set the data source to Kanban, the `dataSource` property is assigned with the instance of the `ej.DataManger`.

**HTML**

```

module KanbanComponent {
$(function () {
var sample = new ej.Kanban($("#Kanban"), {
dataSource: new ej.DataManager(window["kanbanData"]).executeLocal(new
ej.Query().take(20)),
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
]
});
});
}

```

**Mapping Values**

In order to display cards in Kanban control, you need to map the database fields to Kanban cards and columns. The required mapping field are listed as follows

- **keyField** - Map the column name to use as **key** values to columns.
- **columns** - Map the corresponding **key** values of **keyField** column to each columns
- **fields.content** - Map the column name to use as content to cards.
- **fields.primaryKey** - Map the column name to use as primary Key.

**HTML**

```

declare var window:myWindow;
export interface myWindow extends Window{
kanbanData:any;
}
module KanbanComponent {
$(function () {
var sample = new ej.Kanban($("#Kanban"), {
dataSource: new ej.DataManager(<any>window["kanbanData"]).executeLocal(new
ej.Query().take(20)),
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
allowTitle: true,
fields: {

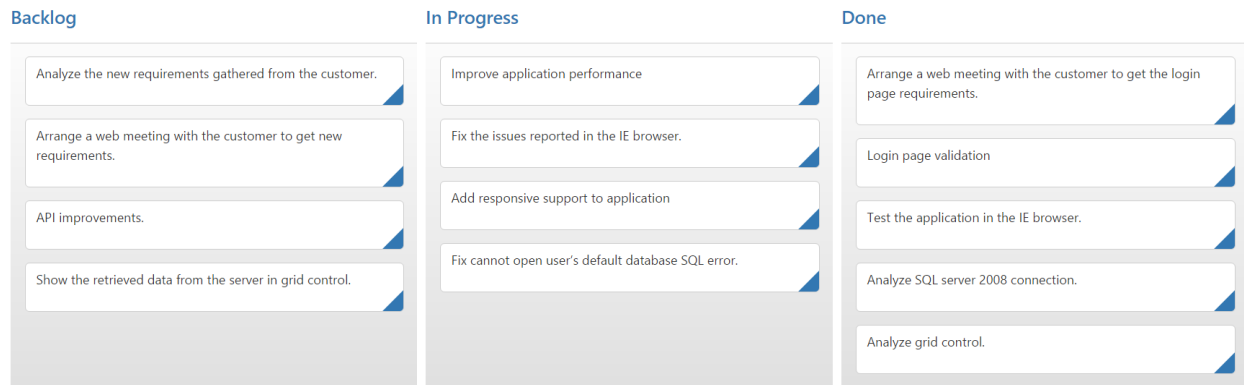
```



```

content: "Summary",
primaryKey: "Id",
}
});
});
}

```



**Note:** `fields.primaryKey` field is mandatory for “Drag and Drop”, “Selection” and “Editing” Features.

### Enable Swimlane

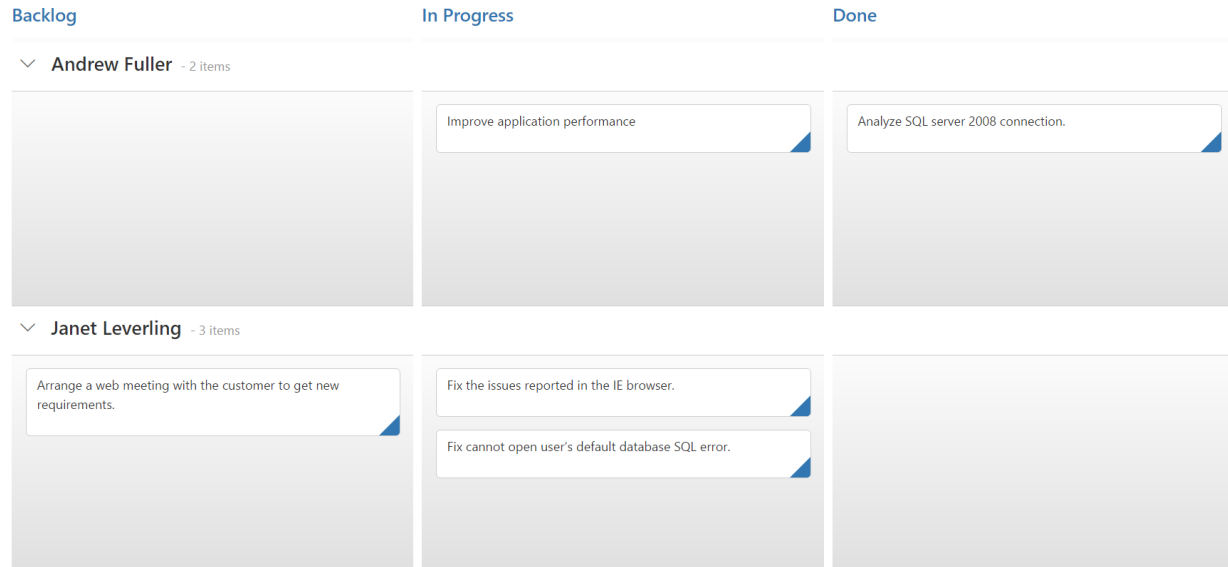
Swimlane can be enabled by mapping the `fields.swimlaneKey` to appropriate column name in `dataSource`. This enables the grouping of the cards based on the mapped column values.

### HTML

```

declare var window:myWindow;
export interface myWindow extends Window{
kanbanData:any;
}
module KanbanComponent {
$(function () {
var sample = new ej.Kanban($("#Kanban"), {
dataSource: new ej.DataManager(<any>window["kanbanData"]).executeLocal(new
ej.Query().take(20)),
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
allowTitle: true,
fields: {
content: "Summary",
swimlaneKey: "Assignee",
primaryKey: "Id",
}
});
});
}

```



### Adding Filters

Filters allows to filter the collection of cards from `dataSource` which meets the predefined `query` in the filters collection. To enable filtering, define `filterSettings` collection with display text and [ej.Query](#).

### HTML

```
declare var window:myWindow;
export interface myWindow extends Window{
  kanbanData:any;
}
module KanbanComponent {
  $(function () {
    var sample = new ej.Kanban($("#Kanban"), {
      dataSource: new ej.DataManager(<any>window["kanbanData"]).executeLocal(new
      ej.Query().take(20)),
      columns: [
        { headerText: "Backlog", key: "Open" },
        { headerText: "In Progress", key: "InProgress" },
        { headerText: "Done", key: "Close" }
      ],
      keyField: "Status",
      fields: {
        content: "Summary",
        primaryKey: "Id",
        swimlaneKey: "Assignee"
      },
      filterSettings: [
        { text: "Janet Issues", query: new ej.Query().where("Assignee", "equal",
        "Janet Leverling") },
        { text: "Closed Issues", query: new ej.Query().where("Status", "equal",
        "Close") }
      ]
    });
  });
}
```

Filters: Janet Issues Closed Issues

## Backlog

## In Progress

## Done

Andrew Fuller - 2 items

Improve application performance

Analyze SQL server 2008 connection.

Janet Leverling - 3 items

Arrange a web meeting with the customer to get new requirements.

Fix the issues reported in the IE browser.

Fix cannot open user's default database SQL error.

## Columns

Column fields are present in the [dataSource](#) schema and it is rendering cards based its mapping column values.

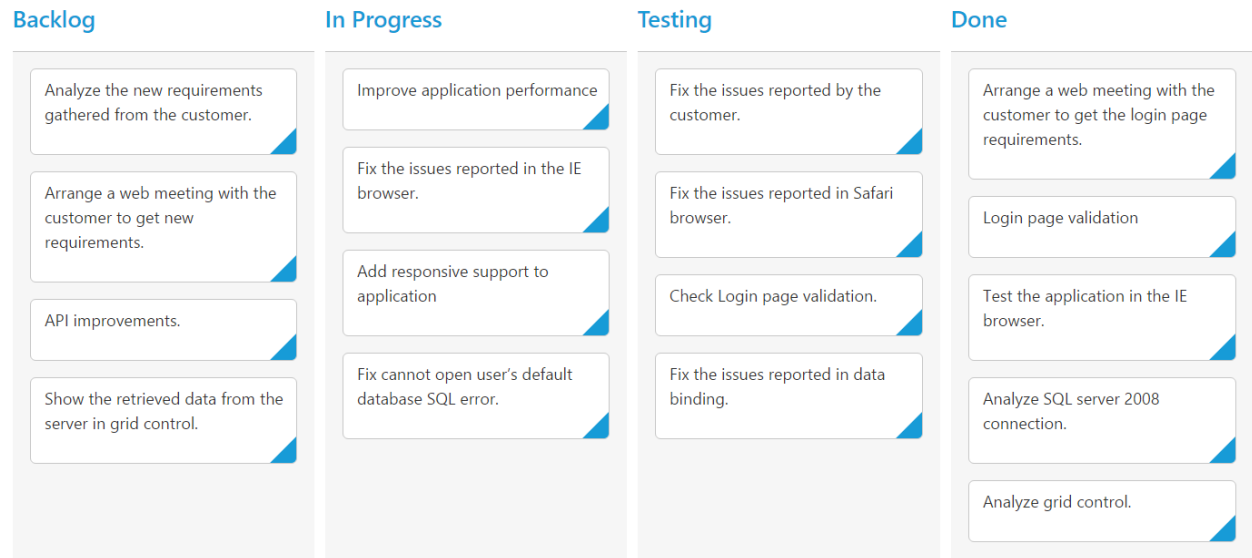
## Key Mapping

To render Kanban with simple cards, you need to map the `dataSource` fields to Kanban cards and [columns](#). The required mapping field are listed as follows

| Mapping Fields                     | Description  |
|------------------------------------|--|
| <a href="#">keyField</a>           | Map the column name to use as <a href="#">key</a> values to columns.                           |
| <a href="#">columns.key</a>        | Map the corresponding <code>key</code> values of <code>keyField</code> column to each columns. |
| <a href="#">columns.headerText</a> | It represents the title for particular column  |
| <a href="#">fields.content</a>     | Map the column name to use as content to cards.  |

**Note:** 1. If the column with `keyField` is not in the `dataSource` and `key` values specified will not available in column values, then the cards will not be rendered.

## 2. If the



fields.content is not in the dataSource, then empty cards will be rendered.

The following code example describes the above behavior.

**HTML**

```
<div id='Kanban'></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open"},
{ headerText: "In Progress", key: "InProgress"},
{ headerText: "Testing", key: "Testing"},
{ headerText: "Done", key: "Close"}
],
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id"
}
});
});
}
```

The following output is displayed as a result of the above code example.



### Multiple Key Mapping

You can map more than one datasource fields as [key](#) values to show different key cards into single column. For e.g , you can map "Validate,In progress" keys under "In progress" column.

The following code example and screenshot which describes the above behavior.

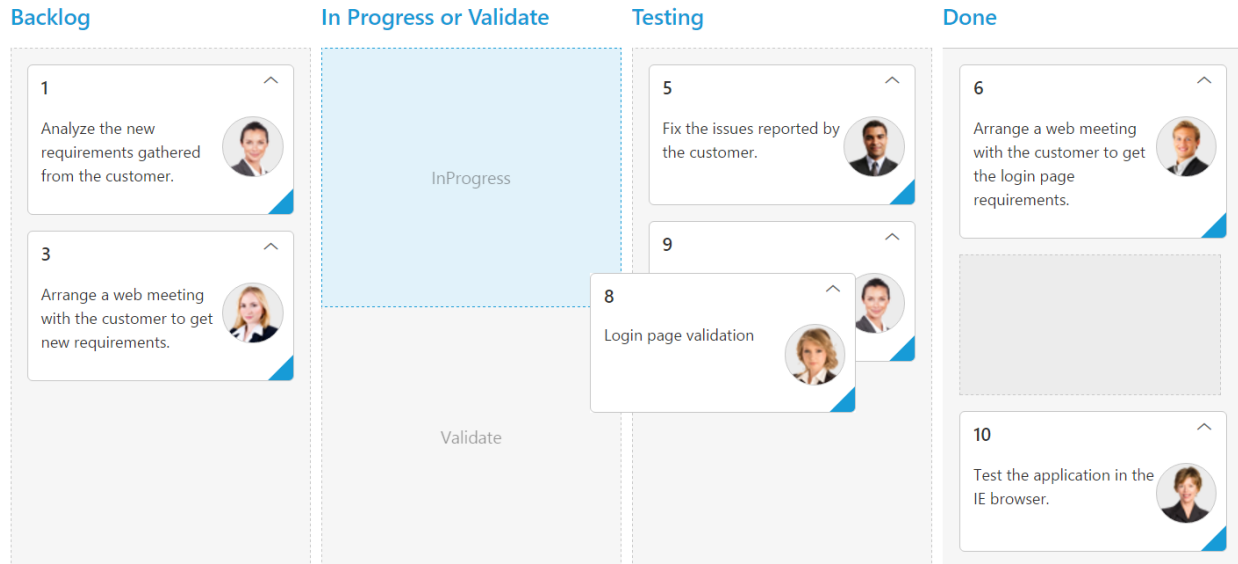
#### HTML

```
<div id='Kanban'></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open"},
{ headerText: "In Progress or Validate", key: "InProgress,Validate" },
{ headerText: "Testing", key: "Testing" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
allowTitle: true,
fields: {
content: "Summary",
primaryKey: "Id"
},
});
});
}
```

The following output is displayed as a result of the above code example.



## Headers

### Header Template

The template design that applies on for the column header. To render template, set [headerTemplate](#) property of the [columns](#).

You can use JsRender syntax in the template. For more information about JsRender syntax, please refer the [link](#).

The following code example describes the above behavior.

### HTML

```
<!--Column Template -->
<script id="column1" type="text/x-jsrender">
<span class="e-backlog e-icon"></span> Backlog
</script>
<div id="column4">
<span class="e-done e-icon"></span> Done
</div>
<div id='Kanban'></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open", headerTemplate: "#column1" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Testing", key: "Testing" },
{ headerText: "Done", key: "Close", headerTemplate: "#column4" }
]
```

```

],
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id"
}
});
});
}

```

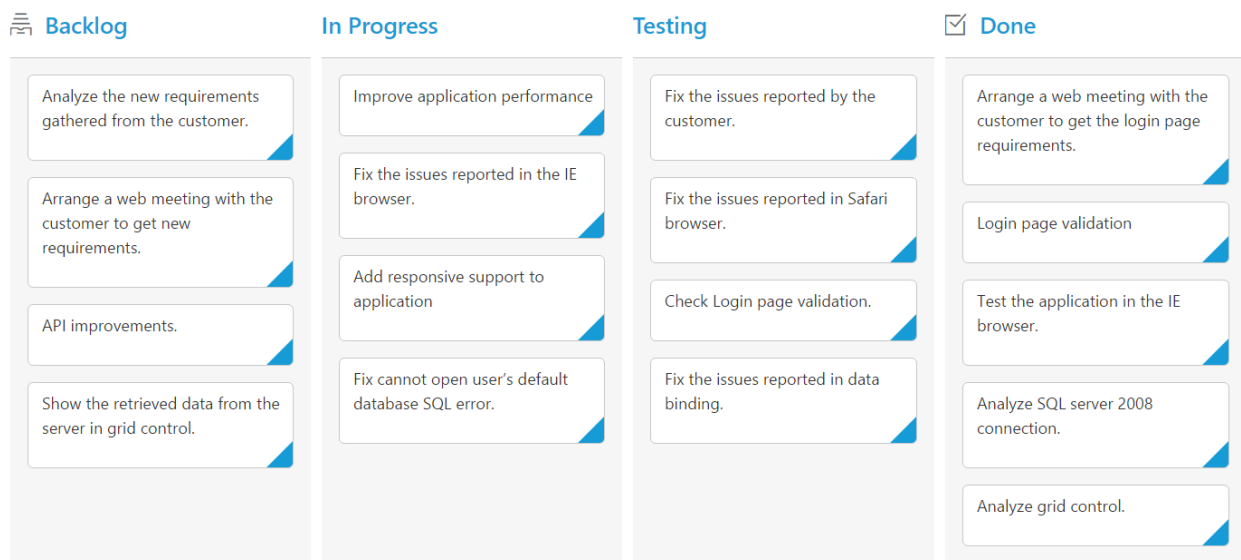
## CSS

```

/*CSS for Header template icon*/
.e-backlog,.e-done {
font-size: 16px;
padding-right: 5px;
display: inline-block;
}
.e-backlog:before {
content: "\e807";
}
.e-done:before {
content: "\e80a";
}

```

The following output is displayed as a result of the above code example.



## Width

You can specify the width for particular column by setting [width](#) property of [columns](#) as in pixel (ex: 100) or in percentage (ex: 40%).

The following code example describes the above behavior.

## HTML

```

<div id='Kanban'></div>

```

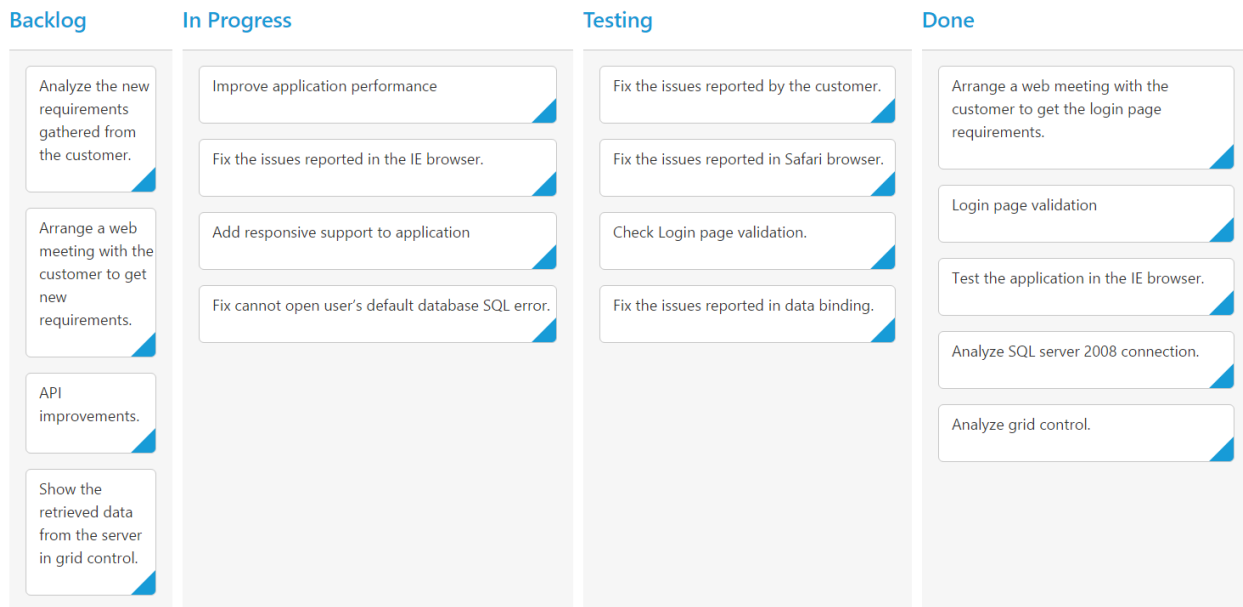
**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open", width: "5%" },
{ headerText: "In Progress", key: "InProgress", width: "12%" },
{ headerText: "Testing", key: "Testing", width: 100 },
{ headerText: "Done", key: "Close", width: 100 }
],
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id"
}
});
});
}

```

The following output is displayed as a result of the above code example.

**Visibility**

You can hide particular column in Kanban by setting [visible](#) property of it as false.

The following code example describes the above behavior.

**HTML**

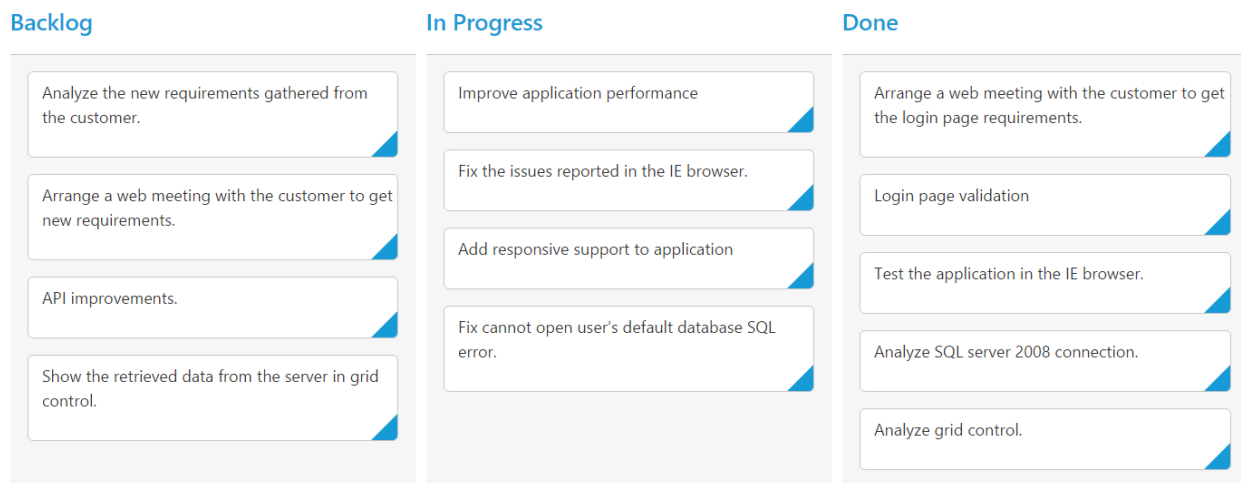


```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Testing", key: "Testing", visible: false },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id"
}
});
});
}
```

The following output is displayed as a result of the above code example.



### Toggle

You can set particular column collapsed state in Kanban by setting [isCollapsed](#) property of it as true. You need to set [allowToggleColumn](#) as true to use “Expand/Collapse” Column.

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

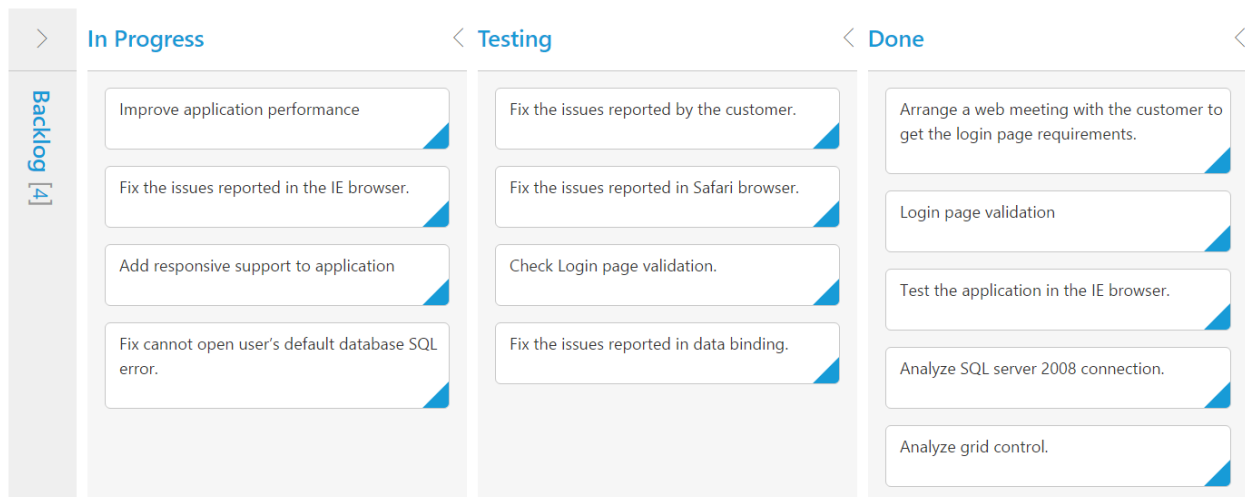
**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
allowToggleColumn:true,
columns: [
{ headerText: "Backlog", key: "Open",isCollapsed:true },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Testing", key: "Testing" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id"
},
});
});
}

```

The following output is displayed as a result of the above code example.

**Allow Dragging**

You can enable and disable drag behavior to the cards in the Kanban columns using the `allowDrag` property and the default value is `true`.

The following code example describes the above behavior.

**HTML**

```
<div id='Kanban'></div>
```

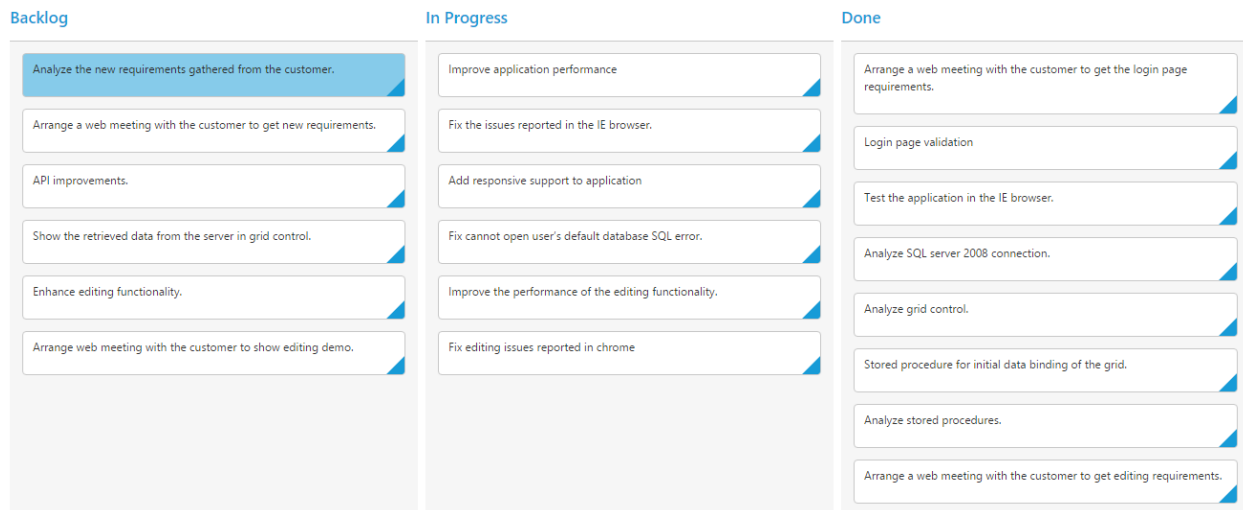
**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open", allowDrag: false },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id",
priority: "RankId"
}
});
});
}

```

The following output is displayed as a result of the above code example.

**Allow Dropping**

You can enable and disable drop behavior to the cards in the Kanban columns using the `allowDrop` property and the default value is `true`.

The following code example describes the above behavior.

**HTML**

```
<div id='Kanban'></div>
```

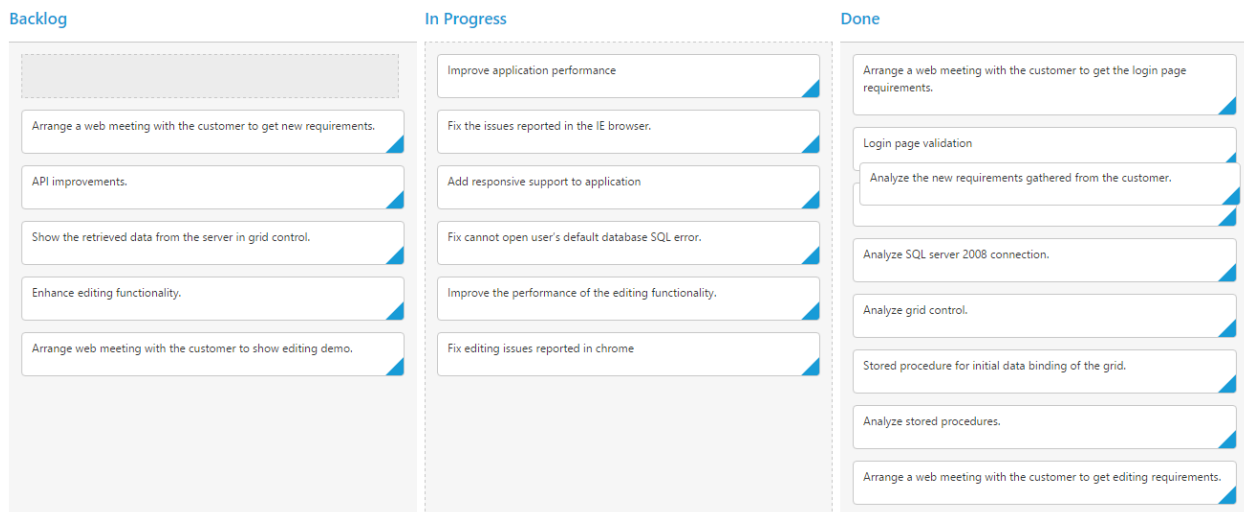
**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close", allowDrop: false }
],
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id",
priority: "RankId"
}
});
});
}

```

The following output is displayed as a result of the above code example.



### Items Count

You can show total cards count in each column's header using the property `enableTotalCount` and the default value is `false`.

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

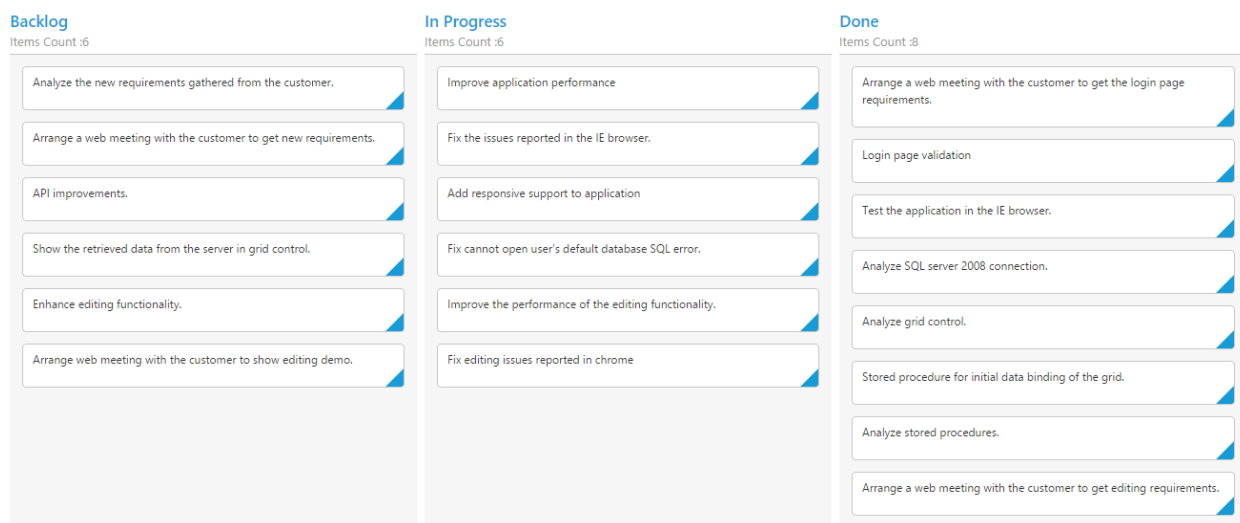
```
/// <reference path="tsfiles/jquery.d.ts" />
```

```

/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
enableTotalCount:true,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id"
}
});
});
}

```

The following output is displayed as a result of the above code example.



### Customize Items Count Text

You can customize the Items count text using the property `totalCount.text`.

The following code example describes the above behavior.

#### HTML

```
<div id='Kanban'></div>
```

#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {

```

```

$(function () {
    var data = new ej.DataManager(window.kanbanData).executeLocal(new
    ej.Query().take(20));
    var sample = new ej.Kanban($("#Kanban"), {
        dataSource: data,
        enableTotalCount: true,
        columns: [
            { headerText: "Backlog", key: "Open", totalCount: {text: "Backlog Count"} },
            { headerText: "In Progress", key: "InProgress" },
            { headerText: "Done", key: "Close" }
        ],
        keyField: "Status",
        fields: {
            content: "Summary",
            primaryKey: "Id"
        }
    });
});

```

The following output is displayed as a result of the above code example.

The screenshot displays a Kanban board with three columns: **Backlog** (Items Count : 6), **In Progress** (Items Count : 6), and **Done** (Items Count : 8). Each column contains a list of task cards. The cards in the 'Backlog' column include tasks like 'Analyze the new requirements gathered from the customer.' and 'Arrange a web meeting with the customer to get new requirements.' The 'In Progress' column shows tasks such as 'Improve application performance' and 'Fix the issues reported in the IE browser.' The 'Done' column lists completed tasks like 'Arrange a web meeting with the customer to get the login page requirements' and 'Login page validation'. Each card has a blue arrow icon in its bottom right corner, indicating a transition to the next stage.

## Data Binding

The Kanban control uses [ej.DataManager](#) which supports both RESTful JSON data services binding and local JSON array binding. The [dataSource](#) property can be assigned either with the instance of [ej.DataManager](#) or JSON data array collection. It supports different kinds of data binding methods such as

1. Local data
2. Remote data

### Local Data

To bind local data to the Kanban, you can assign a JSON array to the [dataSource](#) property.

The JSON array to the [dataSource](#) property can also be provided as an instance of the [ej.DataManager](#). When the JSON array is passed as an instance of [ej.DataManager](#), the [ej.JsonAdaptor](#) will be used to manipulate the Kanban data source.

The following code example describes the above behavior.

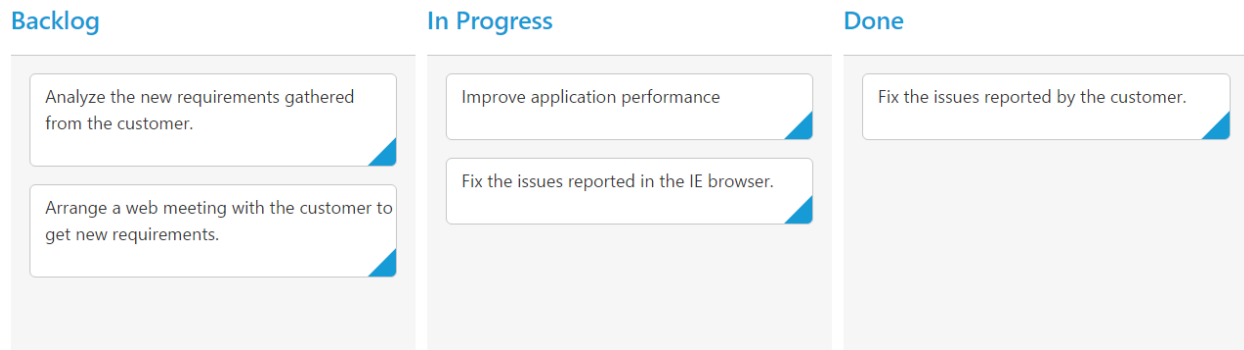
### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
var kanbanData = [
{ Id: 1, Status: "Open", Summary: "Analyze the new requirements gathered
from the customer.", Assignee: "Nancy" },
{ Id: 2, Status: "InProgress", Summary: "Improve application performance",
Assignee: "Andrew" },
{ Id: 3, Status: "Open", Summary: "Arrange a web meeting with the customer
to get new requirements.", Assignee: "Janet" },
{ Id: 4, Status: "InProgress", Summary: "Fix the issues reported in the IE
browser.", Assignee: "Janet" },
{ Id: 5, Status: "Testing", Summary: "Fix the issues reported by the
customer.", Assignee: "Steven" }
];
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: kanbanData,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id"
}
});
});
}
```

The following output is displayed as a result of the above code example.



### Remote Data

To bind remote data to Kanban Control, you can assign a service data as an instance of [ej.DataManager](#) to the [dataSource](#) property.

### OData

OData is a standardized protocol for creating and consuming data. You can provide the [OData service](#) URL directly to the [ej.DataManager](#) class and then you can assign it to Kanban [dataSource](#).

The following code example describes the above behavior.

### HTML

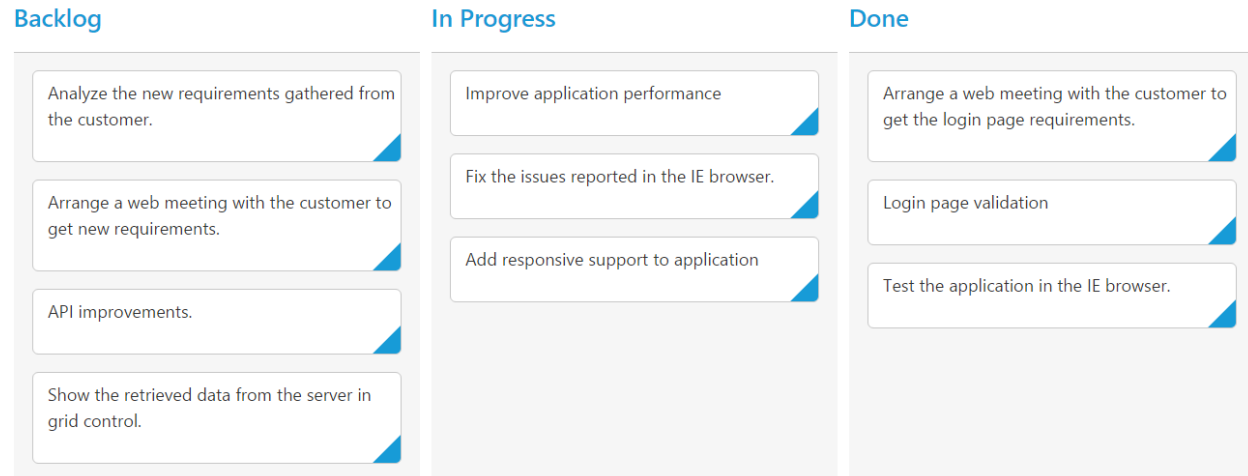
```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
  $(function () {
    var data = new ej.DataManager(window.kanbanData).executeLocal(new
    ej.Query().take(20));
    var sample = new ej.Kanban($("#Kanban"), {
      dataSource: kanbanData,
      columns: [
        { headerText: "Backlog", key: "Open" },
        { headerText: "In Progress", key: "InProgress" },
        { headerText: "Done", key: "Close" }
      ],
      keyField: "Status",
      fields: {
        content: "Summary",
        primaryKey: "Id"
      }
    });
  });
}
```

The following output is displayed as a result of the above code example.





## Workflows

Workflows can be defined to set the flow of card moving between the Kanban column statuses and it is applicable to drag and drop and context menu features.

You can set [workflows](#) as array of Objects which consists of [key](#) and [allowedTransitions](#) properties. The [allowedTransitions](#) accepts more than one transition of the specific column key mentioned in [key](#) property.

If a card is to be dragged to not allowed transition columns, then not supported warning symbol will be displayed for denoting the error.

The following code example describes the above Workflow functionality.

### HTML

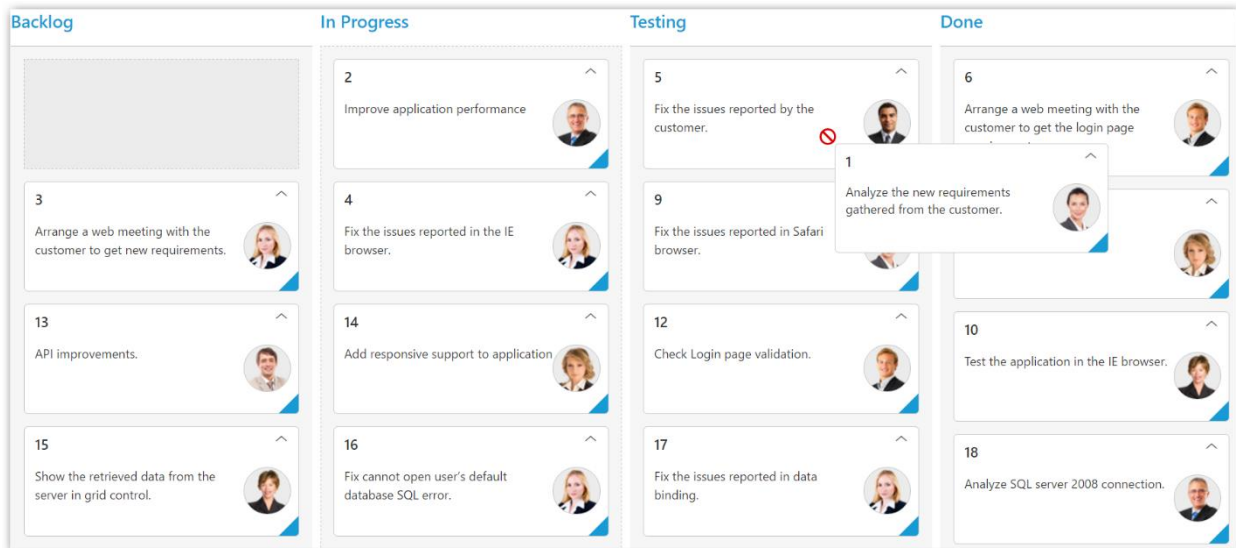
```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var sample = new ej.Kanban($("#Kanban"), {
var data =
ej.DataManager(window.kanbanData).executeLocal(ej.Query().take(30));
$("#Kanban").ejKanban({
dataSource: data,
workflows:[
{ key:"Open",allowedTransitions:"InProgress"},
{ key:"InProgress",allowedTransitions:"Testing,Close"}
],
columns: [
{ headerText: "Backlog", key: "Open"},
{ headerText: "In Progress", key: "InProgress"},
{ headerText: "Testing", key: "Testing"},
{ headerText: "Done", key: "Close"}
],
keyField: "Status",
```

```
fields: {
  content: "Summary"
};
});
});
});
}
```

The following output is displayed as a result of the above code example.



## Swim lanes

Swim lanes are a horizontal categorization of issues in the Kanban control which brings transparency to the workflow. This can be enabled by mapping the [swimlaneKey](#) to appropriate column name in the [dataSource](#).

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
  $(function () {
    var data = new ej.DataManager(window.kanbanData).executeLocal(new
    ej.Query().take(20));
    var sample = new ej.Kanban($("#Kanban"), {
      dataSource: data,
      columns: [
        { headerText: "Backlog", key: "Open" },
        { headerText: "In Progress", key: "InProgress" },
        { headerText: "Done", key: "Close" }
      ],
      keyField: "Status",
```

```
fields: {
  content: "Summary",
  primaryKey: "Id",
  swimlaneKey: "Assignee",
  imageUrl: "ImgUrl"
}
});
});
}
```

The following output is displayed as a result of the above code example.

The Kanban board is organized into three columns: **Backlog**, **In Progress**, and **Done**.

- Backlog:** Contains one card for **Andrew Fuller** (3 items) with the task: "Analyze the new requirements gathered from the customer."
- In Progress:** Contains two cards for **Janet Leverling** (4 items). The first card is "Fix the issues reported by the customer." and the second card is "Add responsive support to application".
- Done:** Contains one card for **Janet Leverling** with the task: "Arrange a web meeting with the customer to get the login page requirements."

## Drag And Drop between swim lanes

You can set '[allowDragAndDrop](#)' property of '[swimlaneSettings](#)' as true to enable Drag and Drop between the swim lanes.

If a card is to be dragged in the same swim lane, only a droppable target cell is added to the dotted line border. If a card is dragged from one swim lane to another, all the Kanban cells will be added to the dotted line borders, except the dragged card cell.

The following code example describes the above behavior.

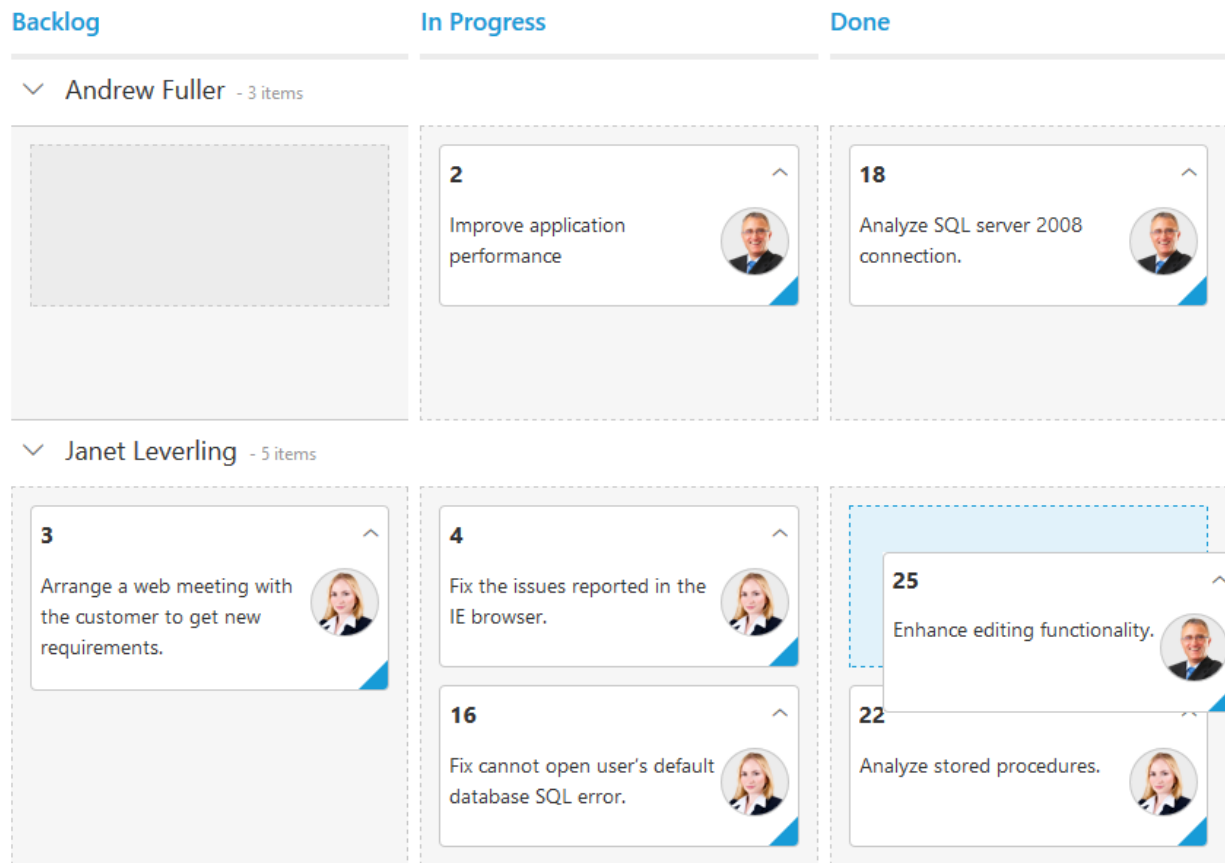
## HTML

<div id='Kanban'></div>

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id",
swimlaneKey: "Assignee",
imageUrl: "ImgUrl"
},
swimlaneSettings:{
allowDragAndDrop: true,
},
});
});
}
```

The following output is displayed as a result of the above code example.



### Unassigned swim lane group

Unassigned swim lane feature provides option to group some common swim lane key values as separate swim lane group. You can enable and disable this behavior using the property [enable](#).

User can use default common key values or user defined key values.

- Using default values
- Using user defined values

**Note:** By default, given common keys are grouped under the swim lane name “Unassigned”, user can customize the name using localization.

### Using default values

By default, the swim lane keys of card which is having null, undefined, empty string ("" ) values will be grouped as unassigned category when [enable](#) property is set as true.

Default values in the [keys](#) collection are null, undefined, empty string ("" ).

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

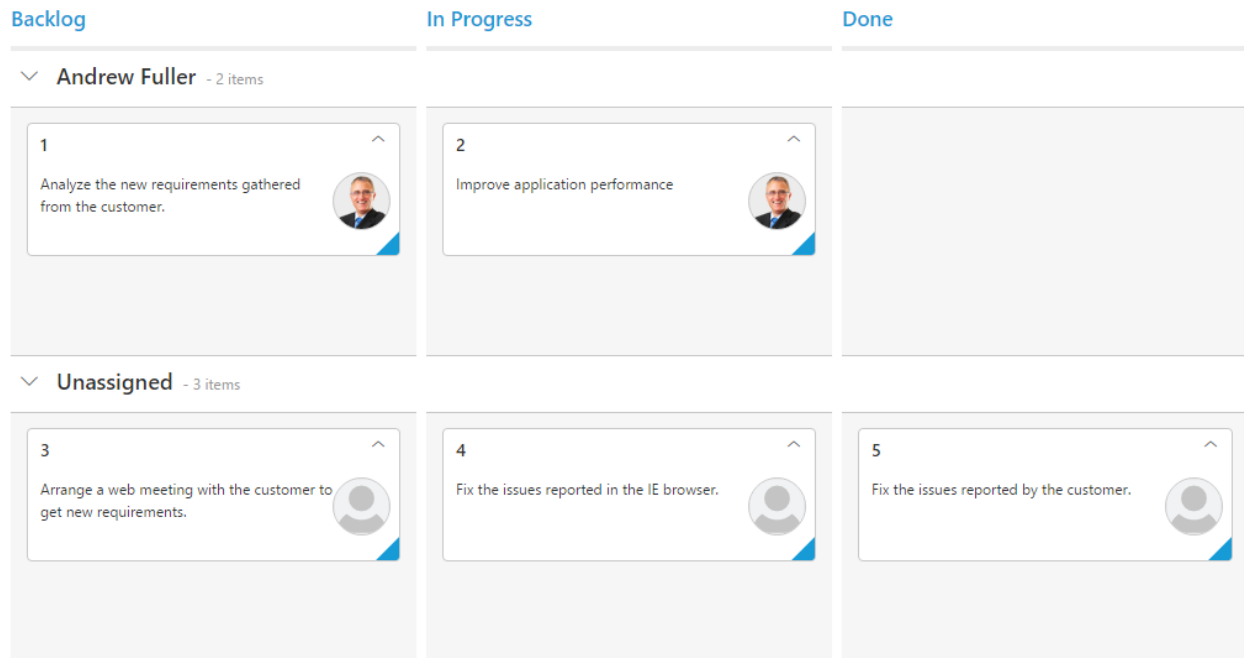
### JAVASCRIPT

```

window.kanbanData = [{ Id: 1, Status: "Open", Summary: "Analyze the new
requirements gathered from the customer.", Type: "Story", Priority: "Low",
Tags: "Analyze, Customer", Estimate: 3.5, Assignee: "Andrew Fuller", ImgUrl:
"../content/images/kanban/2.png", RankId: 1 }, { Id: 2, Status:
"InProgress", Summary: "Improve application performance", Type:
"Improvement", Priority: "Normal", Tags: "Improvement", Estimate: 6,
Assignee: "Andrew Fuller", ImgUrl: "../content/images/kanban/2.png", RankId:
1 }, { Id: 3, Status: "Open", Summary: "Arrange a web meeting with the
customer to get new requirements.", Type: "Others", Priority: "Critical",
Tags: "Meeting", Estimate: 5.5, Assignee: undefined, RankId: 2 }, { Id: 4,
Status: "InProgress", Summary: "Fix the issues reported in the IE browser.",
Type: "Bug", Priority: "Release Breaker", Tags: "IE", Estimate: 2.5,
Assignee: null, RankId: 2 }, { Id: 5, Status: "Close", Summary: "Fix the
issues reported by the customer.", Type: "Bug", Priority: "Low", Tags:
"Customer", Estimate: "3.5", Assignee: "", RankId: 1 }]];
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
allowTitle: true,
fields: {
content: "Summary",
primaryKey: "Id",
swimlaneKey: "Assignee",
imageUrl: "ImgUrl"
},
allowSelection: false
});
});
}

```

The output of the above code example.



### Using user defined values

You can override default values for unassigned swim lane group using the property [keys](#).

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

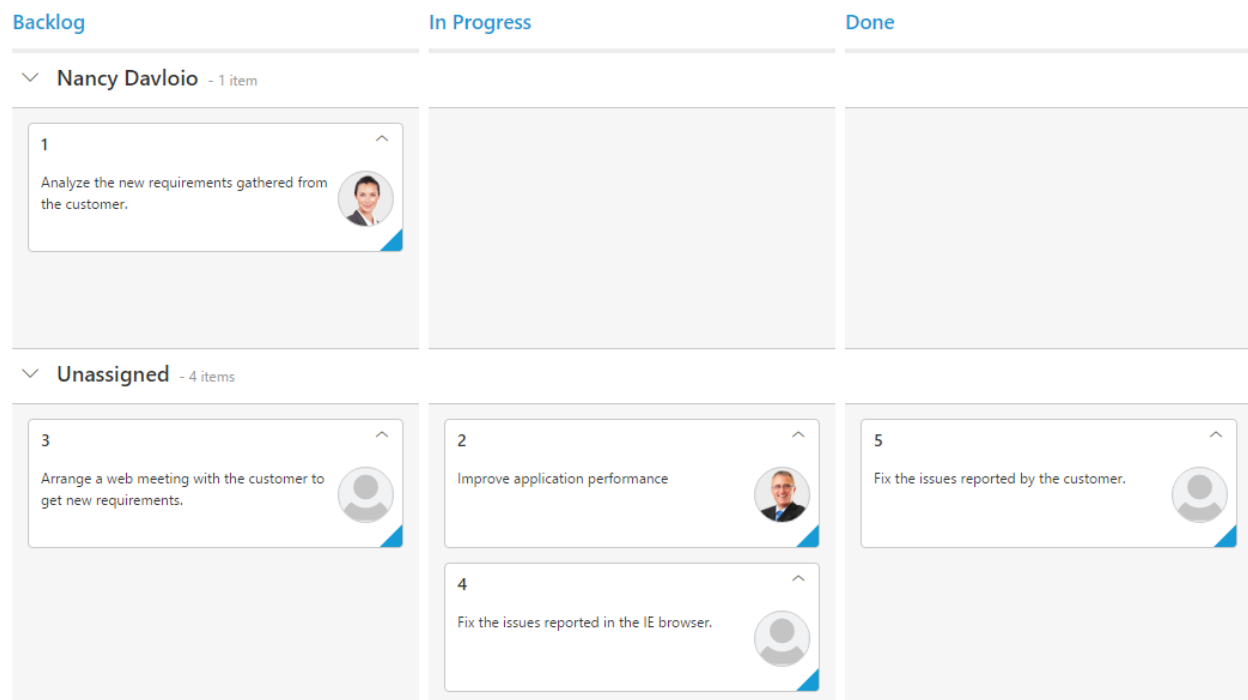
```
window.kanbanData = [
  { Id: 1, Status: "Open", Summary: "Analyze the new requirements gathered
    from the customer.", Type: "Story", Priority: "Low", Tags:
    "Analyze,Customer", Estimate: 3.5, Assignee: "Nancy", ImgUrl: "../content
    /images/kanban/1.png", RankId: 1 },
  { Id: 2, Status: "InProgress", Summary: "Improve application performance",
    Type: "Improvement", Priority: "Normal", Tags: "Improvement", Estimate: 6,
    Assignee: "Andrew", ImgUrl: "../content/images/kanban/2.png", RankId: 1 },
  { Id: 3, Status: "Open", Summary: "Arrange a web meeting with the customer
    to get new requirements.", Type: "Others", Priority: "Critical", Tags:
    "Meeting", Estimate: 5.5, Assignee: "", RankId: 2 },
  { Id: 4, Status: "InProgress", Summary: "Fix the issues reported in the IE
    browser.", Type: "Bug", Priority: "Release Breaker", Tags: "IE", Estimate:
    2.5, Assignee: null, RankId: 2 },
  { Id: 5, Status: "Close", Summary: "Fix the issues reported by the
    customer.", Type: "Bug", Priority: "Low", Tags: "Customer", Estimate: "3.5",
    Assignee: "", RankId: 1 }];
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
  $(function () {
    var data = new ej.DataManager(window.kanbanData).executeLocal(new
    ej.Query().take(20));
```

```

var sample = new ej.Kanban($("#Kanban"), {
  dataSource: data,
  columns: [
    { headerText: "Backlog", key: "Open" },
    { headerText: "In Progress", key: "InProgress" },
    { headerText: "Done", key: "Close" }
  ],
  keyField: "Status",
  allowTitle: true,
  fields: {
    content: "Summary",
    primaryKey: "Id",
    swimlaneKey: "Assignee",
    imageUrl: "ImgUrl"
  },
  swimlaneSettings: {
    unassignedGroup: {
      keys: ["Andrew Fuller", "", "null"]
    }
  },
  allowSelection: false
});

```

The output of the above code example.



### Drag and Drop

By default [allowDragAndDrop](#) is true. Cards can be transited from one column to another column, by dragging and dropping. And it has drop position indicator which enables easier positioning of cards

**Note:** To transit cards to other swim lanes through Drag and Drop, please refer [here](#).



### Prioritization of cards

Prioritizing cards is easy with drag-and-drop re-ordering that helps you to categorize most important work at the top. Cards can be categorized with priority by mapping specific database field into [priority](#) property.

**RankId** defined in the dataSource which is consist priority of cards. The **RankId** will be changed while card ordering changes through **Drag and Drop** and **Editing**.

The following code example describes the above behavior.

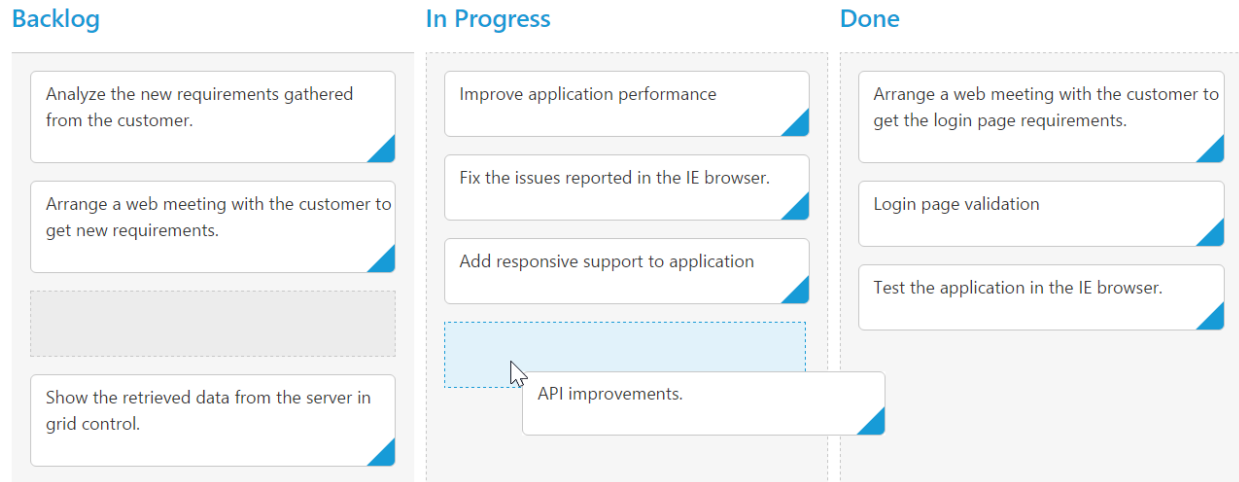
### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id",
priority: "RankId"
}
});
});
}
```

The following output is displayed as a result of the above code example.



**Note:** For Drag and Drop event handling , please refer this [API](#).

## Cards

### Customization

Cards can be customized with appropriate mapping fields from the database. The customizable mapping properties are listed as follows

| Mapping Fields               | Description  |
|------------------------------|--|
| <a href="#">content</a>      | Map the column name to use as content to cards.  |
| <a href="#">tag</a>          | Map the column name to use as tag. Multiple tags can be given with comma separated. E.g. "API", "SQL, Database". |
| <a href="#">color</a>        | Map the column name to use as colors to highlight cards left border.   |
| <a href="#">colorMapping</a> | Map the colors to use with column values which is mapped with <code>fields.color</code> .                        |
| <a href="#">imageUrl</a>     | Map the column name to use as image to cards.  |
| <a href="#">primaryKey</a>   | Map the column name to use as primary key to cards.  |
| <a href="#">priority</a>     | Map the column name to use as priority to cards.   |
| <a href="#">title</a>        | Map the column name to use as title to cards. Default title is <code>primaryKey</code> .                         |
| <a href="#">allowTitle</a>   | Set as true to enable title for card.  |

The following code example describes the above behavior.

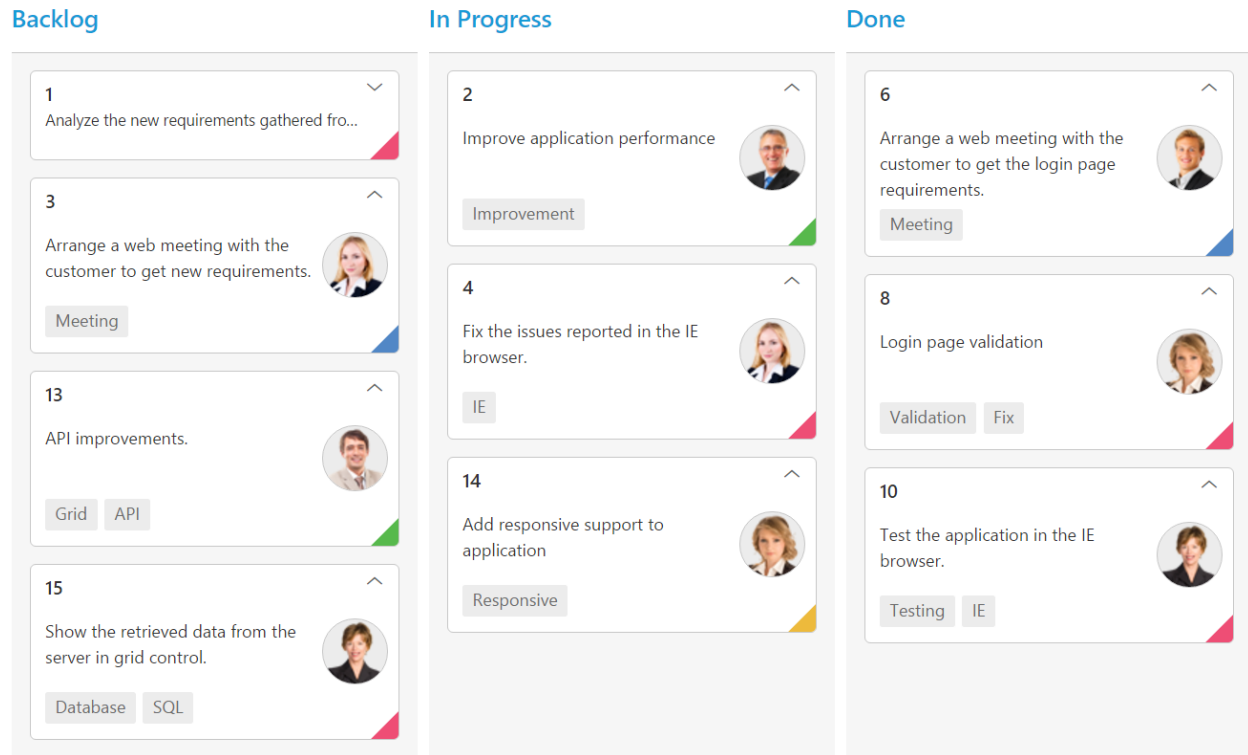
### HTML

```
<div id='Kanban'></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data =
ej.DataManager(window.kanbanData).executeLocal(ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
allowTitle: true,
fields: {
content: "Summary",
primaryKey: "Id",
priority: "RankId",
tag: "Tags",
color: "Type",
imageUrl: "ImgUrl"
},
cardSettings: {
colorMapping: {
"#cb2027": "Bug, Story",
"#67ab47": "Improvement",
"#fbae19": "Epic",
"#6a5da8": "Others"
}
}
});
});
}
```

The following output is displayed as a result of the above code example.



## Template

Templates are used to create custom card layout as per the user convenient. HTML templates can be specified in the [template](#) property of the [cardSettings](#) as an ID of the template's HTML element.

You can use JsRender syntax in the template. For more information about JsRender syntax, please refer this [link](#).

The following code example describes the above behavior.

## HTML

```
<div id='Kanban'></div>
<script id="template" type="text/x-jsrender">
<table>
<tr>
<td class="photo">

</td>
<td class="details">
<table>
<colgroup>
<col width="30%">
<col width="70%">
</colgroup>
<tbody>
<tr>
<td class="CardHeader"> Name: </td>
<td>{{{"{}}":Assignee{{{}}}}</td>
</tr>
<tr>
<td class="CardHeader"> Task: </td>
```

```
<td>{{{"{":Type{}}}}</td>
</tr>
</tbody>
</table>
</td>
</tr>
</table>
</script>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
{
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open"},
{ headerText: "In Progress", key: "InProgress"},
{ headerText: "Done", key: "Close"}
],
keyField: "Status",
fields: {
primaryKey: "Id",
color: "Type",
},
cardSettings: {
template: "#template",
colorMapping: {
"#cb2027": "Bug, Story",
"#67ab47": "Improvement",
"#fbae19": "Epic",
"#6a5da8": "Others"
}
}
});
});
}
```

## CSS

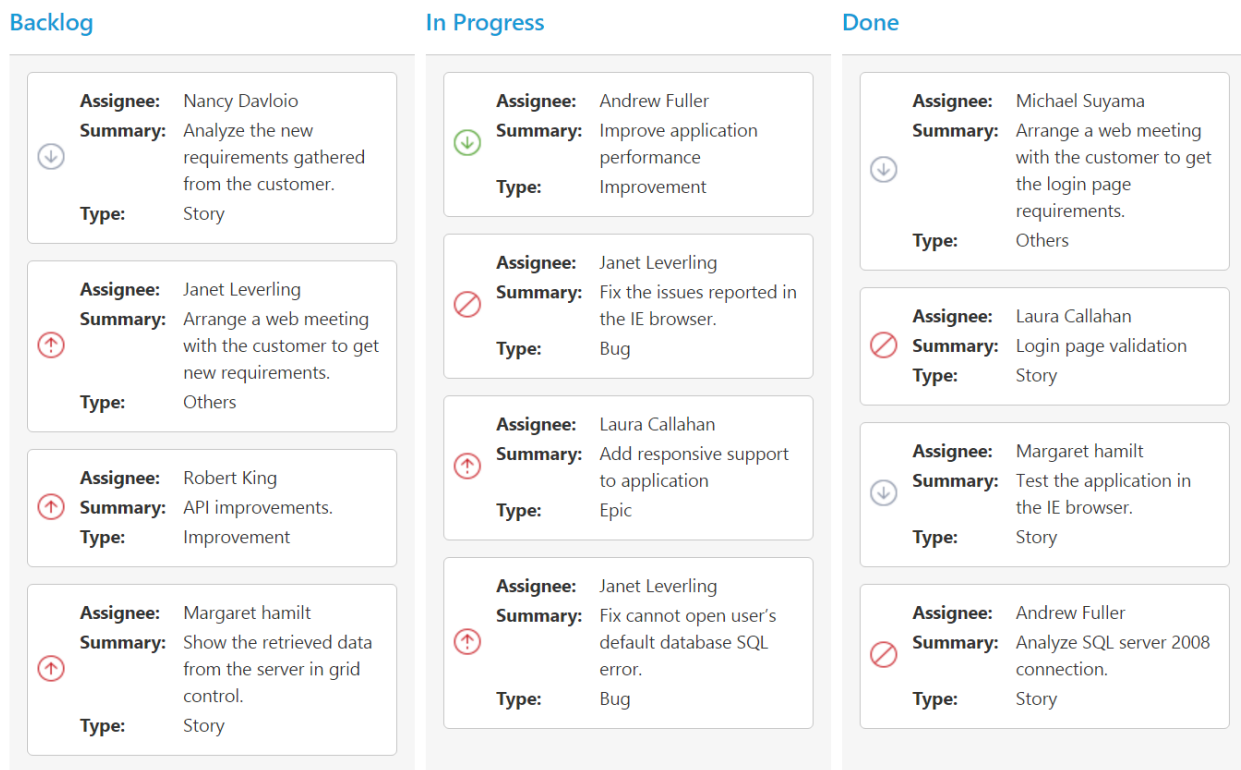
```
<!--CSS for card template-->
<style>
.e-templatetable {
width: 100%;
}
.details > table {
margin-left: 2px;
border-collapse: separate;
border-spacing: 2px;
width: 100%;
}
```

```

.details td {
vertical-align: top;
}
.details {
padding: 8px 8px 10px 0;
}
.photo {
padding: 8px 6px 10px 6px;
text-align: center;
}
.CardHeader {
font-weight: bolder;
padding-right: 10px;
}
</style>

```

The following output is displayed as a result of the above code example.



### Tooltip

You can enable HTML tooltip for Kanban card elements by setting [enable](#) property as true in [tooltipSettings](#).

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

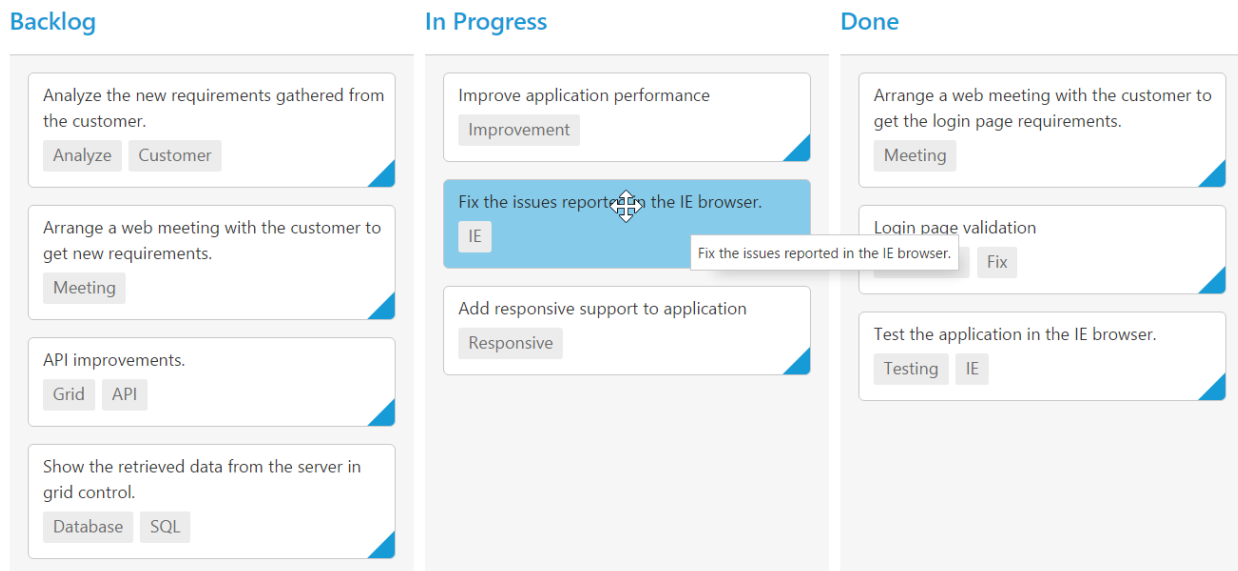
### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
tooltipSettings: {
enable: true
},
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
primaryKey: "Id",
content: "Summary",
tag: "Tags"
}
});
});
}

```

The following output is displayed as a result of the above code example.



### Template

By making use of template feature with tooltip, all the field names that are mapped from the `dataSource` can be accessed to define the [template](#) tooltip for card. The [tooltipSettings.enable](#) must be enabled first.

The following code example describes the tooltip template.

### HTML

```

<div id='Kanban'></div>
<script id="tooltipTemp" type="text/x-jsrender">
<div class='e-kanbantooltiptemplate'>
<table>
<tr>
<td class="photo">

</td>
<td class="details">
<table>
<colgroup>
<col width="30%">
<col width="70%">
</colgroup>
<tbody>
<tr>
<td class="CardHeader">Assignee:</td>
<td>{{ '{' }':Assignee{ } } }</td>
</tr>
<tr>
<td class="CardHeader">Type:</td>
<td>{{ '{' }':Type{ } } }</td>
</tr>
<tr>
<td class="CardHeader">Estimate:</td>
<td>{{ '{' }':Estimate{ } } }</td>
</tr>
<tr>
<td class="CardHeader">Summary:</td>
<td>{{ '{' }':Summary{ } } }</td>
</tr>
</tbody>
</table>
</td>
</tr>
</table>
</div>
</script>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
tooltipSettings: {
enable: true,
template: "#tooltipTemp"
},
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },

```

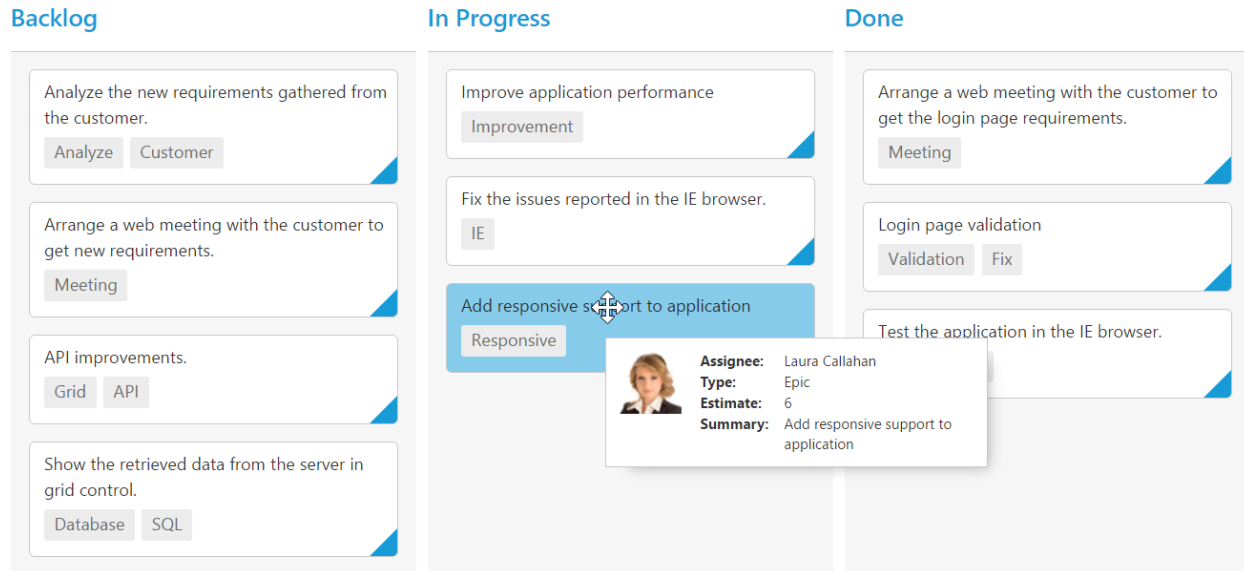


```
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
primaryKey: "Id",
content: "Summary",
tag: "Tags"
}
});
});
}
```

## CSS

```
<!--toolTip template releated css -->
<style>
.details >table {
width: 100%;
margin-left:2px;
border-collapse: separate;
border-spacing: 1px;
}
.e-kanbantooltiptemplate {
width: 250px;
padding: 3px;
}
.e-kanbantooltiptemplate > table {
width: 100%;
}
.e-kanbantooltiptemplate td {
vertical-align: top;
}
td.details {
padding-left: 10px;
}
.CardHeader {
font-weight: bolder;
}
</style>
```

The following output is displayed as a result of the above code example.



### Collapsible Cards

You can set particular cards to the collapsed state in Kanban by defining the [collapsibleCards](#) property. Based on the [collapsibleCards](#) object value, it maps the cards to the collapsible area.

You can set [collapsibleCards](#) as object which consists of [field](#) and [key](#) properties. The [field](#) property map the datasource field to be used in [collapsibleCards](#). The [key](#) property map the specific column key which is to be in collapsed state.

| Mapping Fields                         | Description   |
|--|---|
| <a href="#">collapsibleCards.field</a> | Map the collapsible card's field mapping.   |
| <a href="#">collapsibleCards.key</a>   | Map the collapsible card's key mapping which is available in datasource value of field mapped in <a href="#">collapsibleCards.field</a> . |

**Note:** 1. If the [collapsibleCards](#) with [field](#) is in the [dataSource](#) and [key](#) values specified will available in column values, then the cards will be rendered inside the collapsible card's division.

The following code example describes the collapsible cards.

#### HTML

```
<div id='Kanban'></div>
```

#### JAVASCRIPT

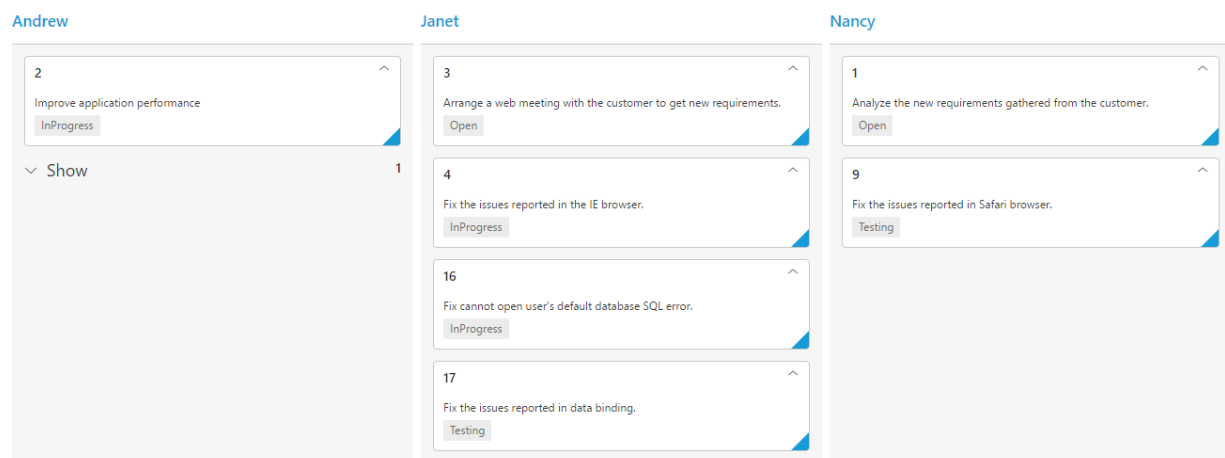
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function) () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
```

```

dataSource: data,
columns: [
{ headerText: "Andrew", key: "Andrew Fuller"},
{ headerText: "Janet", key: "Janet Leverling"},
{ headerText: "Nancy", key: "Nancy Davloio"}
],
keyField: "Assignee",
allowTitle: true,
fields: {
content: "Summary",
primaryKey: "Id",
tag: "Status",
collapsibleCards: { field:"Status", key:"Close"}
},
allowSelection: false,
});
});
}

```

The following output is displayed as a result of the above code example.



**Note:** For cards event handling, please refer this [API](#).

## Editing

The Kanban control has support for dynamic insertion, updating and deletion of cards.

Set [allowEditing](#) and [allowAdding](#) property as true to enable editing/inserting respectively. The primary key for the data source should be defined in [primaryKey](#), for editing to work properly.

You can start the edit action by double clicking the particular card. Similarly, you can add new card to Kanban either by double clicking the particular cell or on an external button which is bound to call [addCard](#) method of Kanban.

Deletion of the card is possible by using [deleteCard](#) by passing primary key as attribute.

**Note:** In Kanban, the **primary key** column will be automatically set to **read only** while editing the card which is to avoid duplicate entry in the cards.

### Configuring Edit Items

You need to configure the list of data source fields that are allowable in editing state using [editItems](#) property. The [field](#) property of [editItems](#) needs to be mapped with data source fields.

You can map the data source field as title to edit form using [title](#) property of [fields](#). By default, it's mapped with [primaryKey](#).

The following code example describes the above behavior.

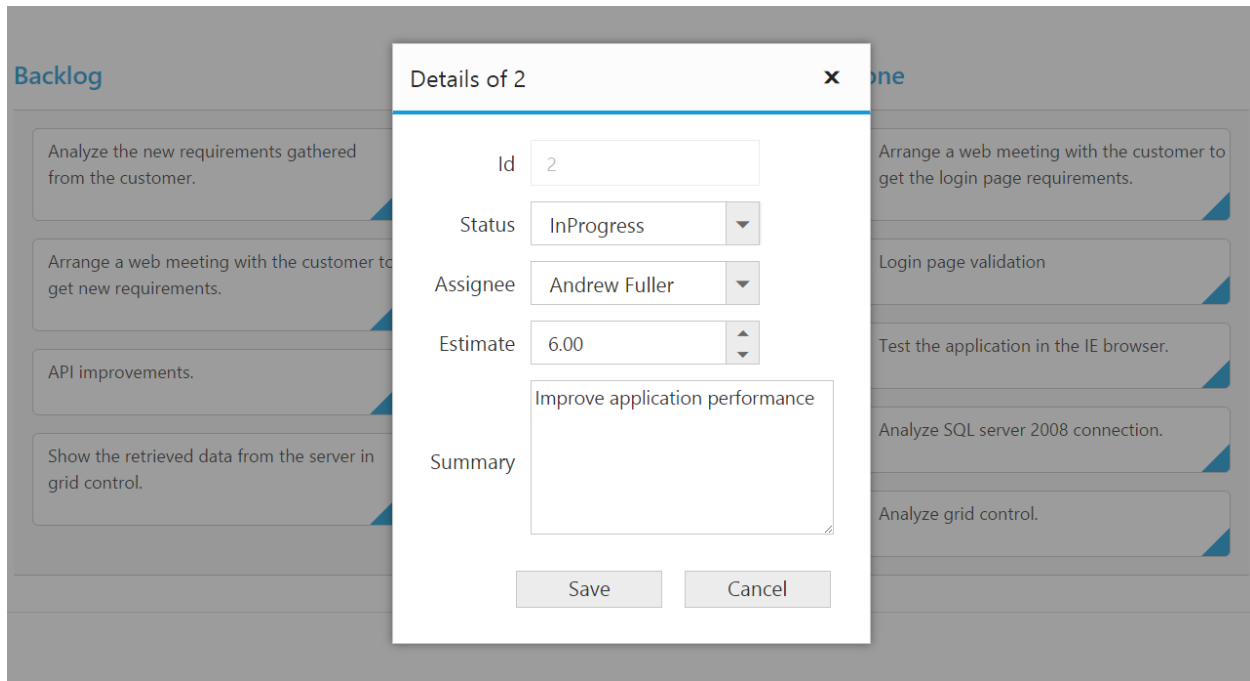
#### HTML

```
<div id='Kanban'></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id"
},
editSettings: {
editItems: [
{ field: "Id" },
{ field: "Status", editType: ej.Kanban.EditingType.Dropdown },
{ field: "Assignee", editType: ej.Kanban.EditingType.Dropdown },
{ field: "Estimate", editType: ej.Kanban.EditingType.Numeric, editParams: {
decimalPlaces: 2 } },
{ field: "Summary", editType: ej.Kanban.EditingType.TextArea, editParams:
{height:100,width:200}}
],
allowEditing: true,
allowAdding: true
}
});
});
}
```

The following output is displayed as a result of the above code example.



**Note:** For editing event handling, please refer this [API](#).

## Edit modes

### Dialog

Set `editMode` as `dialog` to edit data using a dialog box, which displays the fields associated with the data card being edited. Default value is `dialog`.

**Note:** For `editMode` property you can assign either `string` value ("dialog") or `enum` value (`ej.Kanban.EditMode.Dialog`).

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
```

```

content: "Summary",
primaryKey: "Id"
},
editSettings: {
editItems: [
{ field: "Id" },
{ field: "Status", editType: ej.Kanban.EditingType.Dropdown },
{ field: "Assignee", editType: ej.Kanban.EditingType.Dropdown },
{ field: "Estimate", editType: ej.Kanban.EditingType.Numeric, editParams: {
decimalPlaces: 2 } },
{ field: "Summary", editType: ej.Kanban.EditingType.TextArea }
],
allowEditing: true,
allowAdding: true
}
});
});
}

```

The following output is displayed as a result of the above code example.

The screenshot shows a Kanban board interface. In the foreground, a modal dialog titled "Details of 2" is open, allowing for the editing of a specific task card. The dialog contains the following fields:

- Id:** A text input field containing the value "2".
- Status:** A dropdown menu currently set to "InProgress".
- Assignee:** A dropdown menu currently set to "Andrew Fuller".
- Estimate:** A numeric input field with up/down arrows, currently set to "6.00".
- Summary:** A text area containing the text "Improve application performance".

At the bottom of the dialog are "Save" and "Cancel" buttons. The background shows a "Backlog" column with several task cards, including "Analyze the new requirements gathered from the customer.", "Arrange a web meeting with the customer to get new requirements.", "API improvements.", and "Show the retrieved data from the server in grid control.".

#### Dialog Template Form

You can edit any of the fields pertaining to a single card of data and apply it to a template so that the same format is applied to all the other cards that you may edit later.

Using this template support, you can edit the fields that are not bound to [editItems](#).

To edit the cards using Dialog template form, set [editMode](#) as `dialogtemplate` and specify the template id to [dialogTemplate](#) property of [editSettings](#).

**Note:** 1. `value` attribute is used to bind the corresponding field value while editing.

2. `name` attribute is used to get the changed field values while save the edited card.

3. For `editMode` property you can assign either `string` value ("dialogtemplate") or `enum` value (`ej.Kanban.EditMode.DialogTemplate`).

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
<script id="template" type="text/template">
<table cellpadding="10">
<tr>
<td style="text-align: right;">Id
</td>
<td style="text-align: left">
<input id="Id" name="Id" value="{: Id}" class="e-field e-ejinputtext valid
e-disable" style="text-align: right; width: 175px; height: 28px"
disabled="disabled" />
</td>
<td style="text-align: right;">Status
</td>
<td style="text-align: left">
<select id="Status" name="Status">
<option value="Close">Close</option>
<option value="InProgress">InProgress</option>
<option value="Open">Open</option>
</select>
</td>
</tr>
<tr>
<td style="text-align: right;">Estimate
</td>
<td style="text-align: left">
<input type="text" id="Estimate" name="Estimate" value="{: Estimate}" />
</td>
<td style="text-align: right;">Assignee
</td>
<td style="text-align: left">
<select id="Assignee" name="Assignee">
<option value="Nancy">Nancy</option>
<option value="Andrew">Andrew</option>
<option value="Janet">Janet</option>
<option value="Margaret">Margaret</option>
<option value="Steven">Steven</option>
<option value="Michael">Michael</option>
<option value="Robert">Robert</option>
<option value="Laura">Laura</option>
</select>
</td>
</tr>
</table>
</script>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
```

```

$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
actionComplete: "complete",
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id"
},
editSettings: {
allowEditing: true,
allowAdding: true,
editMode: ej.Kanban.EditMode.DialogTemplate,
dialogTemplate: "#template"
},
}
);
});
}

```

While using template, you can change the elements that are defined in the `template`, to appropriate Syncfusion JS controls based on the column type. This can be achieved by using [actionComplete](#) event of Kanban. Please refer to following code snippets.

#### JAVASCRIPT

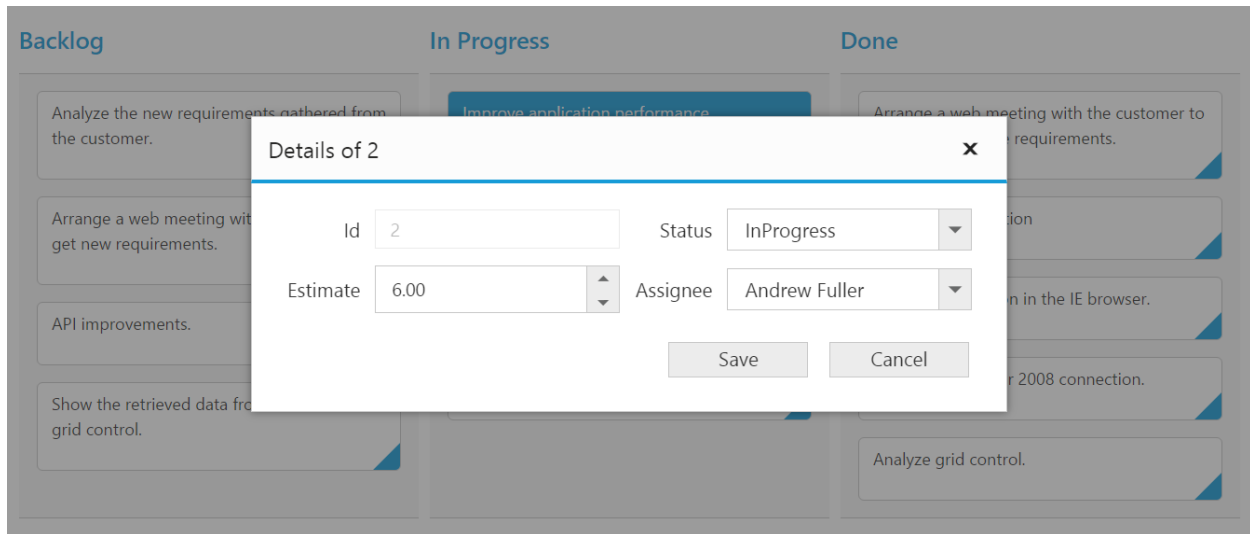
```

function complete(args) {
if ((args.requestType == "beginedit" || args.requestType == "add") &&
args.model.editSettings.editMode == "dialogtemplate") {
$("#Estimate").ejNumericTextbox({ value: parseFloat($("#Estimate").val()),
width: "175px", height: "34px", decimalPlaces: 2 });
$("#Assignee").ejDropDownList({ width: '175px' });
$("#Status").ejDropDownList({ width: '175px' });
if (args.requestType == "beginedit" || args.requestType == "add") {
$("#Assignee").ejDropDownList("setSelectedValue", args.data['Assignee']);
$("#Status").ejDropDownList("setSelectedValue", args.data['Status']);
}
}
}
}

```

The following output is displayed as a result of the above code example.





### External Form

Set the [editMode](#) as externalform to open the edit form in outside kanban content.

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
allowTitle: true,
fields: {
content: "Summary",
primaryKey: "Id"
},
scrollSettings:{
height:500,
width:700,
},
editSettings: {
editMode: ej.Kanban.EditMode.ExternalForm,
editItems: [
{ field: "Id", editType: ej.Kanban.EditingType.String,validationRules: {
required: true, number: true }},

```

```

{ field: "Status", editType: ej.Kanban.EditingType.DropDown },
{ field: "Assignee", editType: ej.Kanban.EditingType.DropDown },
{ field: "Estimate", editType: ej.Kanban.EditingType.Numeric, editParams: {
decimalPlaces: 2 },validationRules: {range: [0, 1000]}}},
{ field: "Summary", editType:
ej.Kanban.EditingType.TextArea,validationRules: { required: true}},
],
allowEditing: true,
allowAdding: true
}
});
});
}

```

The following output is displayed as a result of the above code example.

The screenshot displays a Kanban Board with two columns: 'Backlog' and 'In Progress'. The 'Backlog' column contains three cards: Card 1 (blue header, 'Analyze the new requirements gathered from the customer.'), Card 3 ('Arrange a web meeting with the customer to get new requirements.'), and Card 13 ('API improvements.'). The 'In Progress' column contains two cards: Card 2 ('Improve application performance') and Card 4 ('Fix the issues reported in the IE browser.'). Below the board, a 'Details of 1' modal is open, showing fields for Id (1), Status (Open), and Summary (Analyze the new requirements gathered from the customer.). There are 'Save' and 'Cancel' buttons at the bottom right of the modal.

Form Position:

Form Position can be customized by setting the [formPosition](#) property of [`editSettings`](#) as "right" or "bottom".

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open"},
{ headerText: "In Progress", key: "InProgress"},
{ headerText: "Done", key: "Close"}
],
keyField: "Status",
allowTitle: true,
fields: {
content: "Summary",
primaryKey: "Id"
},
allowScrolling:true,
scrollSettings:{
height:250,
width:700,
},
editSettings: {
editMode:ej.Kanban.EditMode.ExternalForm,
formPosition: ej.Kanban.FormPosition.Bottom,
editItems: [
{ field: "Id", editType: ej.Kanban.EditingType.String,validationRules: {
required: true, number: true }},
{ field: "Status", editType: ej.Kanban.EditingType.Dropdown },
{ field: "Assignee", editType: ej.Kanban.EditingType.Dropdown },
{ field: "Estimate", editType: ej.Kanban.EditingType.Numeric, editParams: {
decimalPlaces: 2 },validationRules: {range: [0, 1000]}},
{ field: "Summary", editType:
ej.Kanban.EditingType.TextArea,validationRules: { required: true}}
],
allowEditing: true,
allowAdding: true
},
});
});
}
```

The following output is displayed as a result of the above code example.

The screenshot shows a Kanban board with two columns: 'Backlog' and 'In Progress'. The 'Backlog' column contains four cards with IDs 1, 3, 13, and 15. The 'In Progress' column contains three cards with IDs 2, 4, and 14. A modal titled 'Details of 2' is open, displaying the details for card ID 2. The modal fields are: Id (2), Status (InProgress), Assignee (Andrew Fuller), Estimate (6.00), and Summary (Improve application performance). There are 'Save' and 'Cancel' buttons at the bottom of the modal.

### External Template Form

You can edit any of the fields pertaining to a single card of data and apply it to a template so that the same format is applied to all the other cards that you may edit later.

Using this template support, you can edit the fields that are not bound to Kanban Edit Items.

To edit the cards using External template form, set `editMode` as `externalformtemplate` and specify the template id to `externalFormTemplate` property of `editSettings`.

While using template, you can change the elements that are defined in the template, to appropriate Syncfusion JS controls based on the column type. This can be achieved by using `actionComplete` event of Kanban.

**Note:** 1. `value` attribute is used to bind the corresponding field value while editing.

2. `name` attribute is used to get the changed field values while save the edited card.

3. For `editMode` property you can assign either `string` value ("externalformtemplate") or `enum` value (`ej.Kanban.EditMode.ExternalFormTemplate`).

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
<script id="template" type="text/template">
<table cellpadding="10">
<tr>
<td style="text-align:left;">Id
</td>
<td style="text-align: left">
<input id="Id" name="Id" value="{:{: Id}}" class="e-field e-ejinputtext valid
e-disable" style="text-align: right; width: 175px; height: 28px"
disabled="disabled" />
</td>
</tr>
</table>
```

```

<td style="text-align: left;">Status
</td>
<td style="text-align: left">
<select id="Status" name="Status">
<option value="Close">Close</option>
<option value="InProgress">InProgress</option>
<option value="Open">Open</option>
<option value="Testing">Testing</option>
<option value="Validate">Validate</option>
</select>
</td>
</tr>
<tr>
<td style="text-align: left;">Assignee
</td>
<td style="text-align: left">
<select id="Assignee" name="Assignee">
<option value="Nancy">Nancy</option>
<option value="Andrew">Andrew</option>
<option value="Janet">Janet</option>
<option value="Margaret">Margaret </option>
<option value="Steven">Steven</option>
<option value="Michael ">Michael</option>
<option value="Robert">Robert</option>
<option value="Laura">Laura</option>
</select>
</td>
</tr>
<tr>
<td style="text-align: left;">Priority
</td>
<td style="text-align: left">
<input id="Priority" name="Priority" value="{{: Priority}}" class="e-field
e-ejinputtext valid" style="width: 175px; height: 28px" />
</td>
</tr>
<tr>
<td style="text-align: left;">Summary
</td>
<td style="text-align: left">
<textarea id="Summary" name="Summary" class="e-ejinputtext" value="{{:
Summary}}" style="width: 270px; height: 95px">{{: Summary}}</textarea>
</td>
</tr>
</table>
</script>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {

```

```
dataSource: data,
actionComplete: "complete",
columns: [
{ headerText: "Backlog", key: "Open"},
{ headerText: "In Progress", key: "InProgress"},
{ headerText: "Done", key: "Close"}
],
keyField: "Status",
allowTitle: true,
fields: {
content: "Summary",
primaryKey: "Id"
},
allowScrolling: true,
scrollSettings: {
height: 450,
width: 700,
},
editSettings: {
editMode: ej.Kanban.EditMode.ExternalFormTemplate,
externalFormTemplate: "#template",
allowEditing: true,
allowAdding: true
},
};
});
}

function complete(args) {
if ((args.requestType == "beginedit" || args.requestType == "add") &&
args.model.editSettings.editMode == "externalformtemplate") {
$("#Assignee").ejDropDownList({ width: '175px' });
$("#Status").ejDropDownList({ width: '175px' });
if(args.requestType == "beginedit" || args.requestType == "add" ){
$("#Assignee").ejDropDownList("setSelectedValue", args.data['Assignee']);
$("#Status").ejDropDownList("setSelectedValue", args.data['Status']);
}
}
}
```

The following output is displayed as a result of the above code example.

### Backlog

1

Analyze the new requirements gathered from the customer.

^

3

Arrange a web meeting with the customer to get new requirements.

^

13

API improvements.

^

### In Progress

2

Improve application performance

^

4

Fix the issues reported in the IE browser.

^

14

Add responsive support to application

^

Details of 1

Id

1

Status

Open

▼

Assignee

Nancy Davloio

▼

Save

Cancel

### Cell edit type and its params

The edit type of bound column can be customized using [editType](#) property of [editItems](#). The following Essential JavaScript controls are supported built-in by [editType](#). And also you can define the model for all the edit types controls while editing through [editParams](#) property of [editItems](#).

The following table describes [editType](#) and their corresponding [editParams](#) of the specific data type of the column.

| EditType       | EditParams                       | Description                                      | Example                             |
|----------------|----------------------------------|--|-------------------------------------|
| Numeric        | <a href="#">ejTextBoxes</a>      | control for integers, double, and decimal data™s | editParams: { decimalPlaces: 2 }    |
| String         | HTML Textbox                     | HTML Textbox                                     | -                                   |
| DatePicker     | <a href="#">ejDatePicker</a>     | control for date data                            | editParams: { buttonText : "Now" }  |
| DateTimePicker | <a href="#">ejDateTimePicker</a> | control for date data-time data                  | editParams: { enabled: true }       |
| DropDown       | <a href="#">ejDropDownList</a>   | control for list of data                         | editParams: { allowGrouping: true } |

|          |                       |  |                                   |
|----------|-----------------------|--|-----------------------------------|
| RTE      | <a href="#">ejRTE</a> | control for customizing text in RTE format | editParams: { allowResize: true } |
| TextArea | HTML TextArea         | Control for multi-line plain-text editing  | editParams:{height:100,width:200} |

**Note:** 1. If [editType](#) is not set, then by default it will display HTML textbox while editing a card.

2. For [editType](#) property you can assign either string value ("numericedit") or **enum** value (ej.Kanban.EditingType.Numeric).

The following code example describes the above behavior.

### HTML

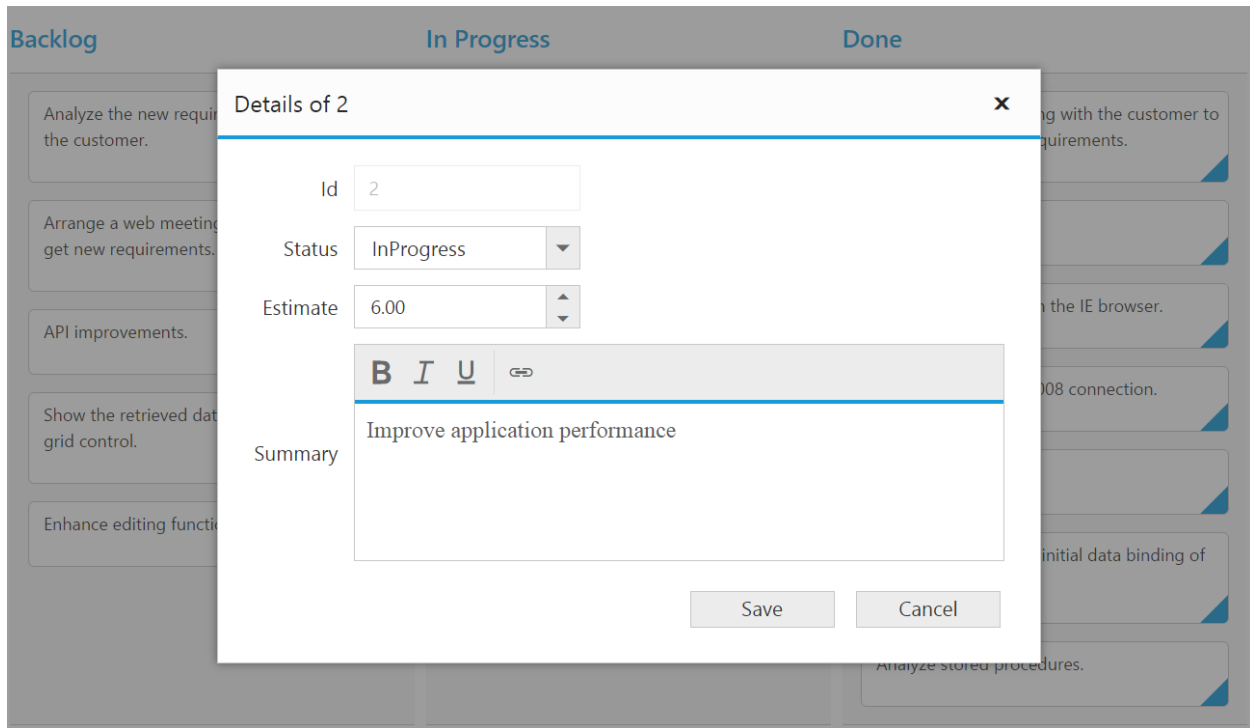
```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
    $(function () {
        var data = new ej.DataManager(window.kanbanData).executeLocal(new
        ej.Query().take(20));
        var sample = new ej.Kanban($("#Kanban"), {
            dataSource: data,
            columns: [
                { headerText: "Backlog", key: "Open" },
                { headerText: "In Progress", key: "InProgress" },
                { headerText: "Done", key: "Close" }
            ],
            keyField: "Status",
            fields: {
                content: "Summary",
                primaryKey: "Id"
            },
            editSettings: {
                editItems: [
                    { field: "Id" },
                    { field: "Status", editType: ej.Kanban.EditingType.Dropdown },
                    { field: "Estimate", editType: ej.Kanban.EditingType.Numeric, editParams: {
                        decimalPlaces: 2 } },
                    { field: "Summary", editType: ej.Kanban.EditingType.RTE, editParams: {
                        height:150,minHeight: 100 } }
                ],
                allowEditing: true,
                allowAdding: true
            }
        });
    });
}
```

The following output is displayed as a result of the above code example.





### Column Validation

We can validate the value of the added or edited card cell before saving.

The below validation script files are needed when editing is enabled with validation.

1. jquery.validate.min.js
2. jquery.validate.unobtrusive.min.js

**Note:** If you enabled the unobtrusive option, then need to refer the jquery.validate.unobtrusive.min.js file in your application along with the other script.

### jQuery Validation

You can set validation rules using [validationRules](#) property of [columns](#). The following are jQuery validation methods.

#### List of jQuery validation methods

| Rules       | Description  |
|-------------|--|
| required    | Requires an element.                                   |
| remote      | Requests a resource to check the element for validity. |
| minlength   | Requires the element to be of given minimum length.    |
| maxlength   | Requires the element to be of given maximum length.    |
| rangelength | Requires the element to be in given value range.       |
| min         | The element requires a given minimum.                  |
| max         | The element requires a given maximum.                  |
| range       | Requires the element to be in a given value range.     |

|            |  |
|------------|--|
| email      | The element requires a valid email.              |
| url        | The element requires a valid url.                |
| date       | Requires the element to be a date.               |
| dateISO    | The element requires an ISO date.                |
| number     | The element requires a decimal number.           |
| digits     | The element requires digits only.                |
| creditcard | Requires the element to be a credit card number. |
| equalTo    | Requires the element to be the same as another.  |

Kanban supports all the standard validation methods of jQuery, please refer the jQuery validation documentation [link](#) for more information.

The following code example describes the above behavior.

#### HTML

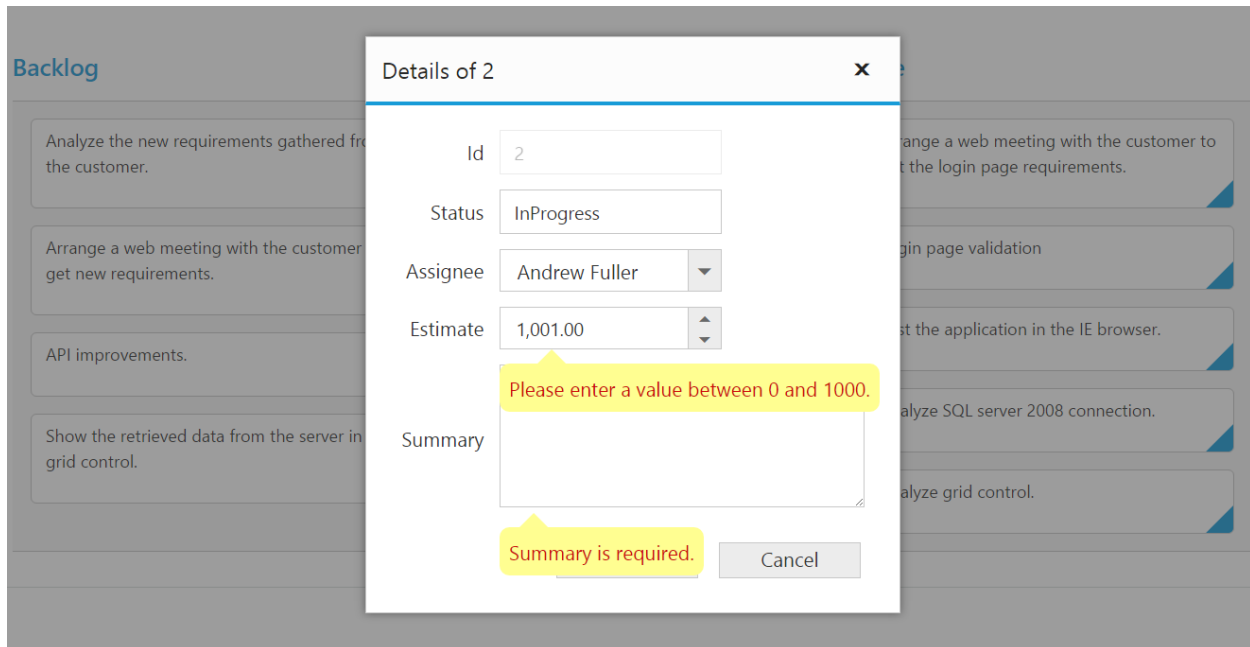
```
<div id='Kanban'></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
primaryKey: "Id",
content: "Summary"
},
editSettings: {
editItems: [
{ field: "Id" },
{ field: "Status", editType: ej.Kanban.EditingType.String },
{ field: "Assignee", editType: ej.Kanban.EditingType.Dropdown },
{ field: "Estimate", editType: ej.Kanban.EditingType.Numeric, editParams: {
decimalPlaces: 2 }, validationRules: { range: [0, 1000] } },
{ field: "Summary", editType: ej.Kanban.EditingType.TextArea,
validationRules: { required: true } }
],
allowEditing: true,
```

```
allowAdding: true
}
});
});
}
```

The following output is displayed as a result of the above code example.



## Filtering

Filtering allows to filter the collection of cards from `dataSource` which meets the predefined `query` in the quick filters collection. To enable filtering, define `filterSettings` collection with display `text` and `ej.Query`.

You can also define display tip to describe filter definition to user using property `description`. If the `description` property is not defined, given `text` will act as display tip.

We can also customize filter option through external button or `customToolbarItems` by using `filterCards()` method.

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

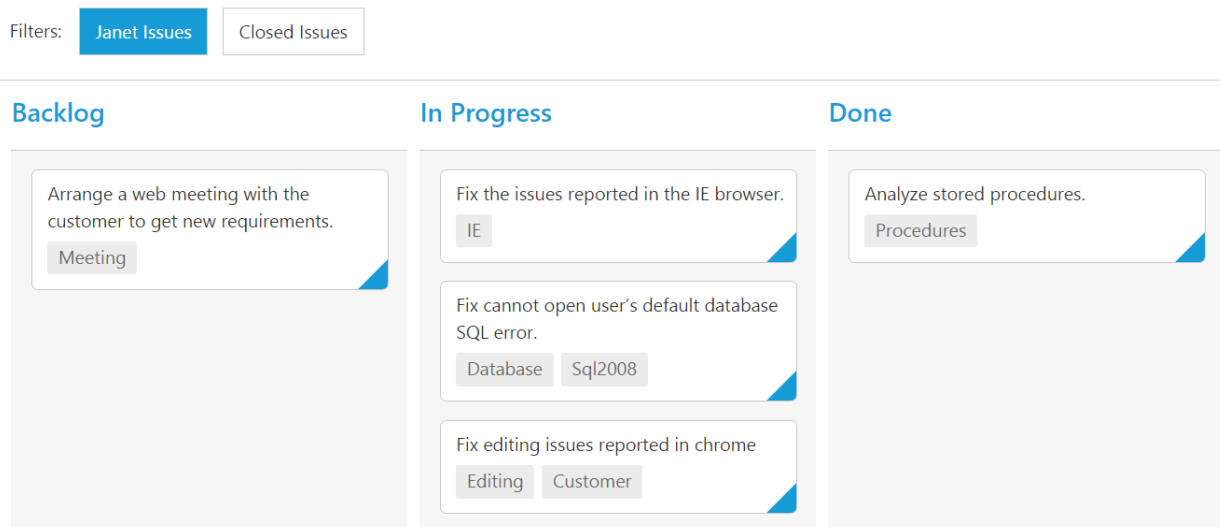
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
```

```

dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
primaryKey: "Id",
content: "Summary",
tag: "Tags"
},
filterSettings: [
{ text: "Janet Issues", query: new ej.Query().where("Assignee", "equal", "Janet"), description: "Displays issues which matches the assignee as 'Janet'" },
{ text: "Closed Issues", query: new ej.Query().where("Status", "equal", "Close"), description: "Display the 'Closed' issues" }
]
});
});
}

```

The following output is displayed as a result of the above code example.



## Scrolling

Scrolling can be enabled by setting [allowScrolling](#) as true. The height and width can be set to Kanban by using the properties [scrollSettings.height](#) and [scrollSettings.width](#) respectively.

**Note:** The height and width can be set in percentage and pixel. The default value for [height](#) and [width](#) in [scrollSettings](#) is 0 and auto respectively.

### Set width and height in pixel

To specify the [scrollSettings.width](#) and [scrollSettings.height](#) in pixel, by set the pixel value as integer.

The following code example describes the above behavior.

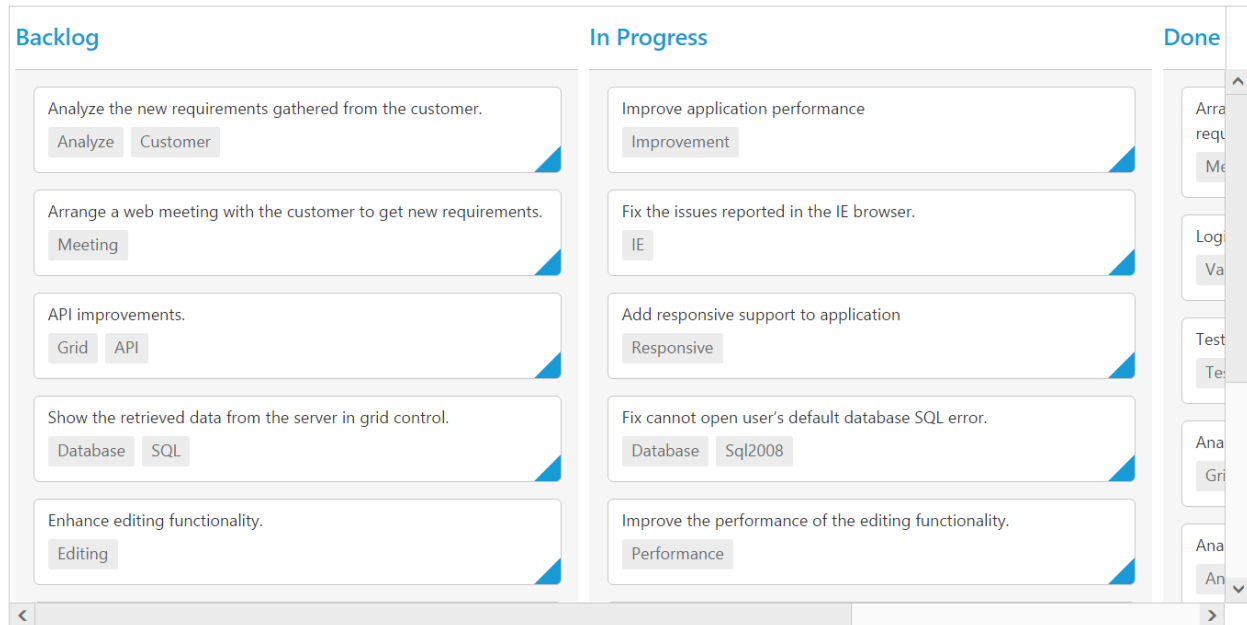
## HTML

```
<div id='Kanban'></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
allowScrolling: true,
scrollSettings: {
width: 900,
height: 450
},
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
primaryKey: "Id",
content: "Summary",
tag: "Tags"
}
});
});
}
```

The following output is displayed as a result of the above code example.



Set height and width in percentage

To specify the [scrollSettings.width](#) and [scrollSettings.height](#) in percentage, by set the percentage value as string.

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

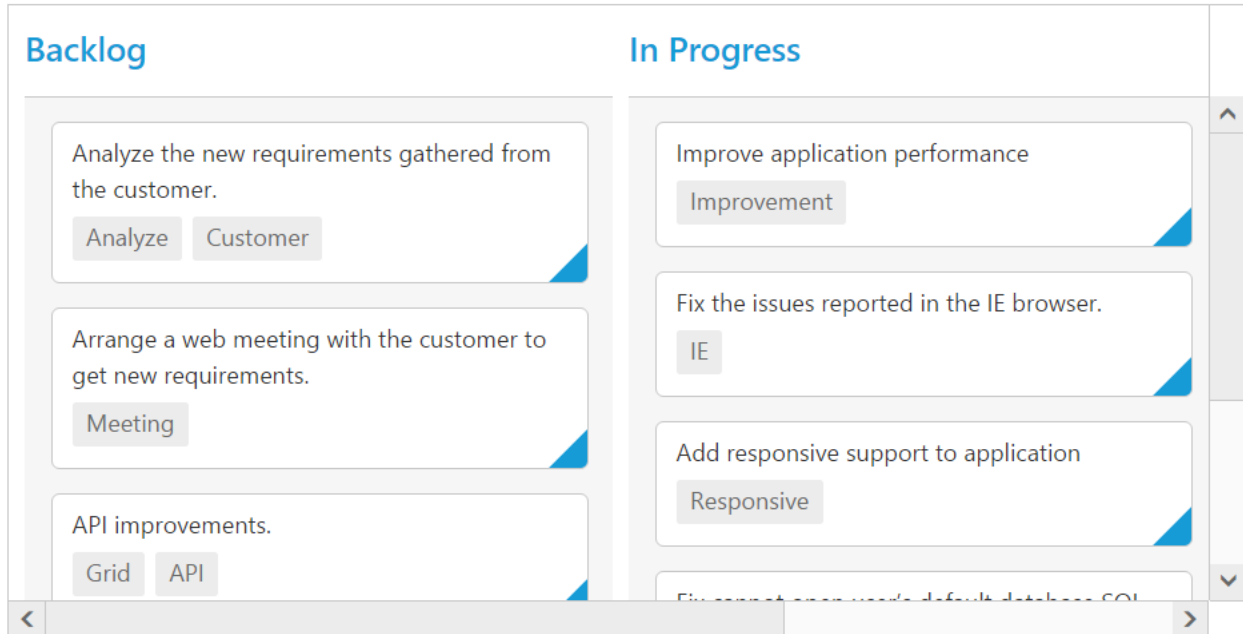
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
    $(function () {
        var data = new ej.DataManager(window.kanbanData).executeLocal(new
        ej.Query().take(20));
        var sample = new ej.Kanban($("#Kanban"), {
            dataSource: data,
            allowScrolling: true,
            scrollSettings: {
                width: "70%", height: "70%"
            },
            columns: [
                { headerText: "Backlog", key: "Open" },
                { headerText: "In Progress", key: "InProgress" },
                { headerText: "Done", key: "Close" }
            ],
            keyField: "Status",
            fields: {
                primaryKey: "Id",
                content: "Summary",
                tag: "Tags"
            }
        })
    })
}
```

```

});
});
}

```

The following output is displayed as a result of the above code example.



Set width as auto

Specify [width](#) property of [scrollSettings](#) as auto, then the scrollbar is rendered only when the Kanban width exceeds the browser window width.

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

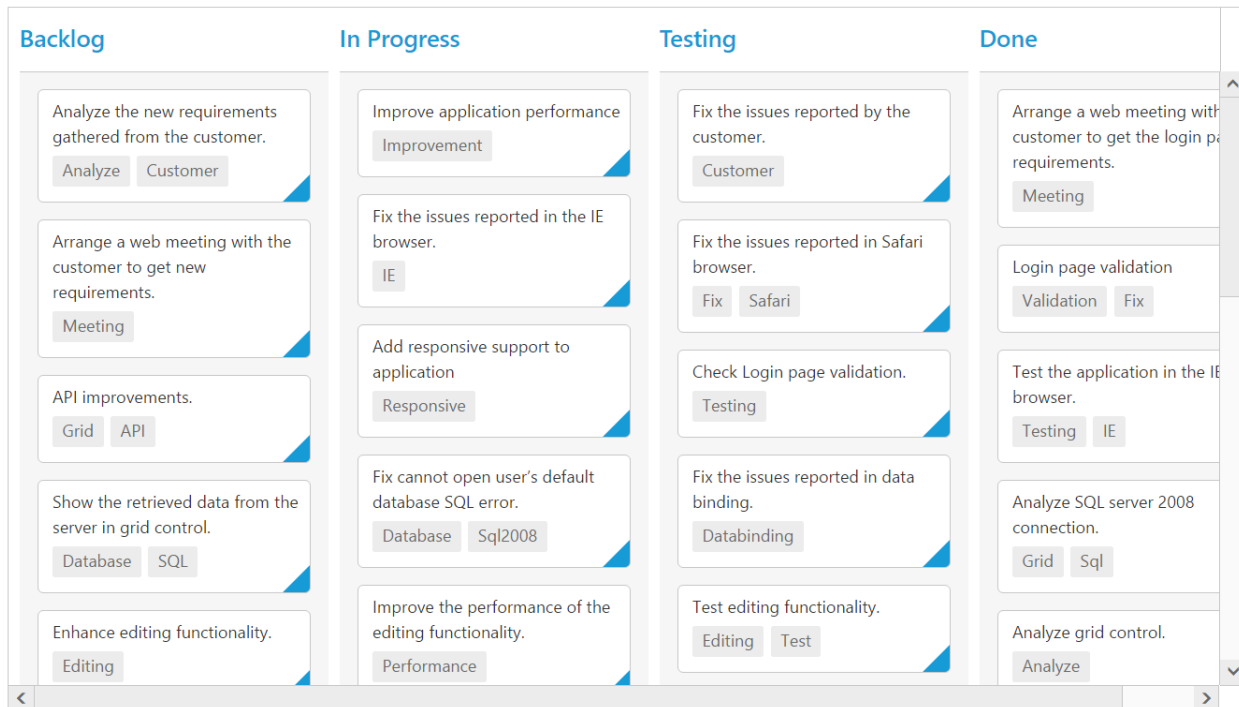
```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
allowScrolling: true,
scrollSettings: { width: "auto", height: 500 },
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Testing", key: "Testing" },
{ headerText: "Done", key: "Close" }
],

```

```
keyField: "Status",
fields: {
content: "Summary",
primaryKey: "Id",
tag: "Tags"
}
});
});
}
```

The following output is displayed as a result of the above code example.



### Enabling freeze swim lane

Set [allowFreezeSwimlane](#) as true. This enables scrolling with freezing of swim lane until you scroll to the next Swim lane, which helps user to aware of current swim lane target.

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
```

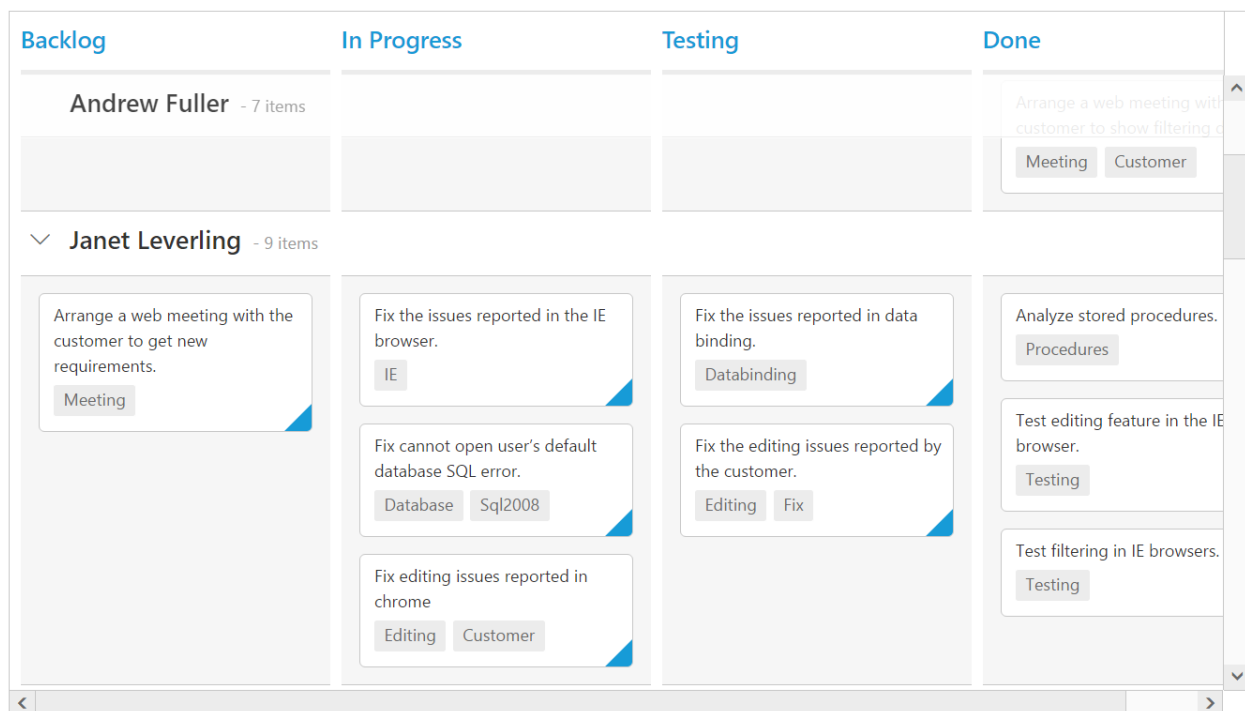


```

allowScrolling: true,
scrollSettings: { width: "auto", height: 500, allowFreezeSwimlane: true },
columns: [
  { headerText: "Backlog", key: "Open" },
  { headerText: "In Progress", key: "InProgress" },
  { headerText: "Testing", key: "Testing" },
  { headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
  content: "Summary",
  primaryKey: "Id",
  swimlaneKey: "Assignee",
  tag: "Tags"
}
});
});
}

```

The following output is displayed as a result of the above code example.



**Note:** `allowFreezeSwimlane` is applicable when swim lane grouping enabled by setting `swimlaneKey`.

### Selection and Hovering

Selection provides an interactive support to highlight the card that you select. Selection can be done through simple Mouse down or Keyboard interaction. To enable selection, set [allowSelection](#) as true.

You can see the mouse hovering effect on the corresponding cards using [allowHover](#) property. By default selection and hovering is `true`.

## Types of Selection

Two types of selections available in Kanban are,

1. Single
2. Multiple

### Single Selection

To enable single selection by setting [selectionType](#) property as single.

### Multiple Selection

Multiple selections is an interactive support to select a group of cards in Kanban by mouse or keyboard interactions. To enable multiple selections by set [selectionType](#) property as `multiple`.

You can select multiple random cards below key press.

| Keys               | Description                      |
|--------------------|----------------------------------|
| Ctrl + mouse left  | To select multiple random cards. |
| Shift + mouse left | To continuous cards select.      |

To unselect selected cards, by press "Shift + mouse left" click on selected row.

The following code example describes the above behavior.

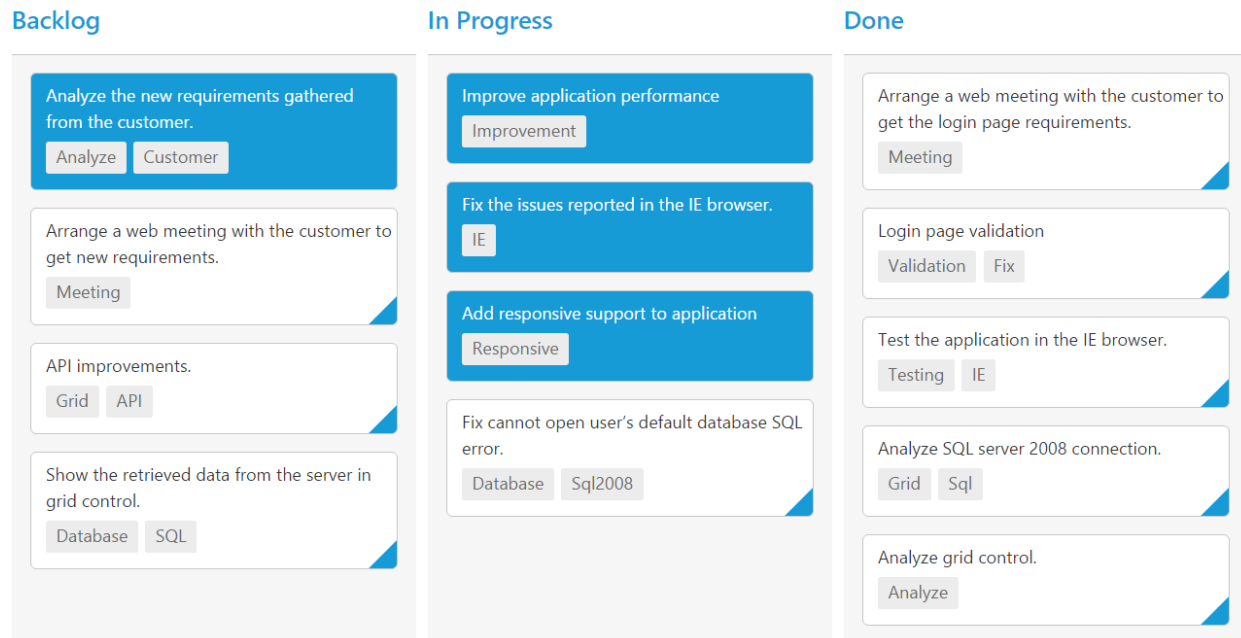
### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
    $(function () {
        var data = new ej.DataManager(window.kanbanData).executeLocal(new
        ej.Query().take(20));
        var sample = new ej.Kanban($("#Kanban"), {
            dataSource: data,
            columns: [
                { headerText: "Backlog", key: "Open" },
                { headerText: "In Progress", key: "InProgress" },
                { headerText: "Done", key: "Close" }
            ],
            keyField: "Status",
            fields: {
                primaryKey: "Id",
                content: "Summary",
                tag: "Tags"
            },
            selectionType: ej.Kanban.SelectionType.Multiple
        });
    });
}
```

The following output is displayed as a result of the above code example.

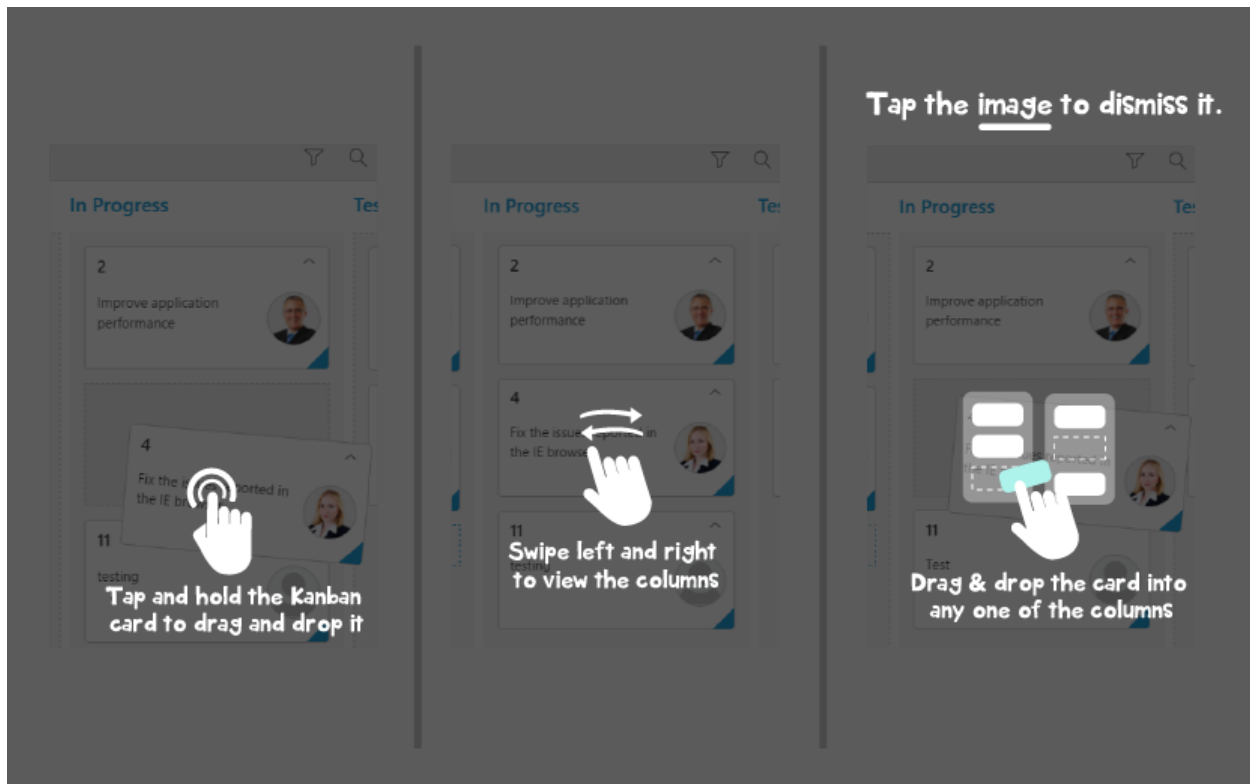


## Responsive

The Kanban control has support for responsive behavior based on client browser's width and height. To enable responsive, [isResponsive](#) property should be true. There are two modes of responsive layout is available in Kanban based on client width. They are.

- Mobile(<480px)
- Desktop(>480px)

You can check the image representation of touch actions from the below image.



### Mobile Layout

If client width is less than 480px, the Kanban will render in mobile mode. In which, you can see that kanban user interface is customized and redesigned for best view in small screens. To enable responsive, [isResponsive](#) property should be true.

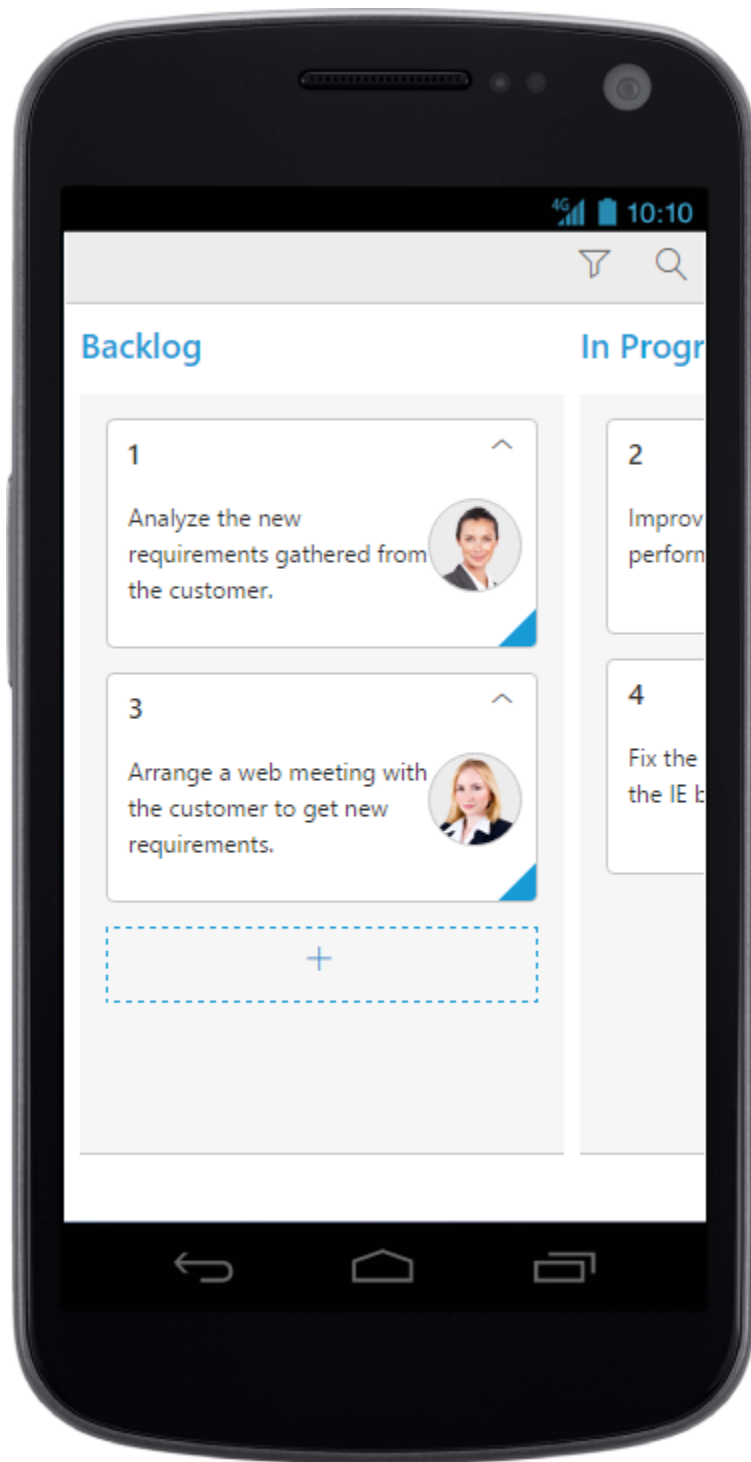
### HTML

```
<div id='Kanban'></div>
```

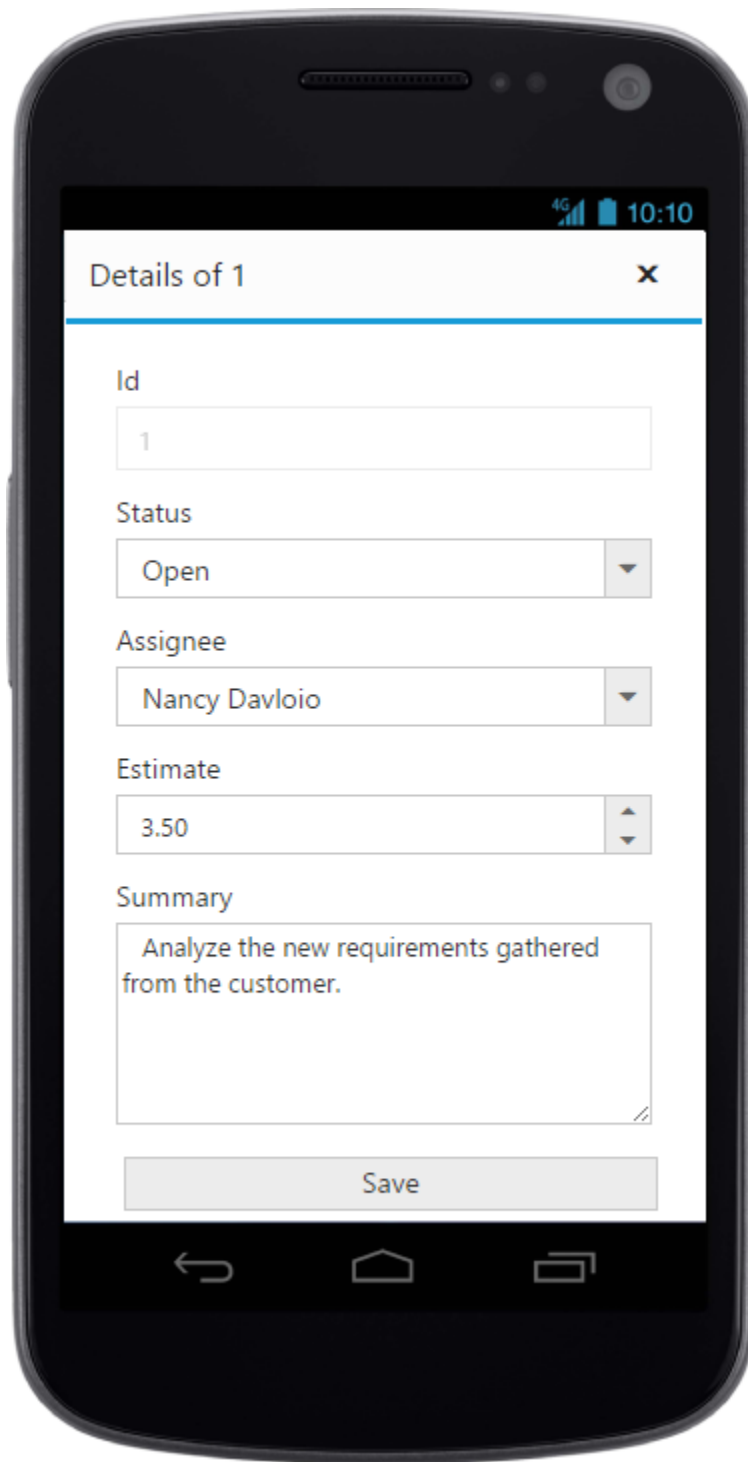
### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
isResponsive: true,
allowSelection: false,
allowKeyboardNavigation: true,
allowTitle: true,
columns: [
{ headerText: "Backlog", key: "Open", showAddButton: true },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Testing", key: "Testing" },
{ headerText: "Done", key: "Close" }
],
editSettings: {
```

```
editItems: [
  { field: "Id", editType: ej.Kanban.EditingType.String, validationRules: {
    required: true, number: true } },
  { field: "Status", editType: ej.Kanban.EditingType.Dropdown },
  { field: "Assignee", editType: ej.Kanban.EditingType.Dropdown },
  { field: "Estimate", editType: ej.Kanban.EditingType.Numeric, editParams: {
    decimalPlaces: 2 }, validationRules: { range: [0, 1000] } },
  { field: "Summary", editType: ej.Kanban.EditingType.TextArea,
    validationRules: { required: true } }
],
allowEditing: true,
allowAdding: true
},
keyField: "Status",
allowSearching: true,
filterSettings: [
  { text: "Janet Issues", query: new ej.Query().where("Assignee", "equal",
    "Janet"), description: "Displays issues which matches the assignee as
    'Janet'" },
  { text: "Testing Issues", query: new ej.Query().where("Status", "equal",
    "Testing"), description: "Display the issues of 'Testing'" }
],
fields: {
  primaryKey: "Id",
  content: "Summary",
  imageUrl: "ImgUrl"
}
});
});
}
```



**Warning:** IE8 and IE9 does not support responsive kanban. `ej.responsive.css` should be referred to display Responsive Kanban.

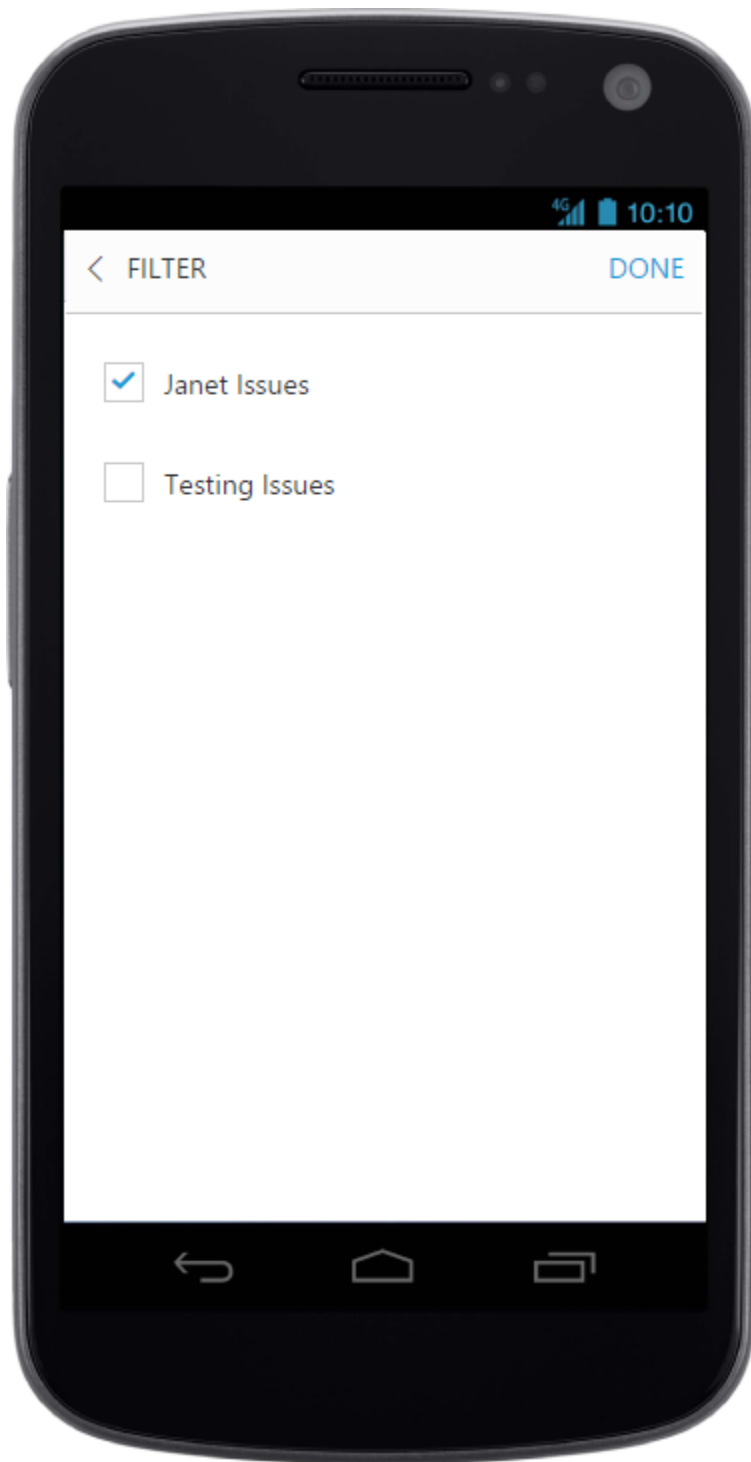


The image shows a mobile application interface on a smartphone. At the top, the status bar displays '4G', signal strength, battery level, and the time '10:10'. The app's header is 'Details of 1' with a close button (X) on the right. The form contains the following fields:

- Id:** A text input field containing the value '1'.
- Status:** A dropdown menu currently showing 'Open'.
- Assignee:** A dropdown menu currently showing 'Nancy Davloio'.
- Estimate:** A numeric input field containing '3.50' with up and down arrow controls.
- Summary:** A text area containing the text 'Analyze the new requirements gathered from the customer.' with a small edit icon at the bottom right.

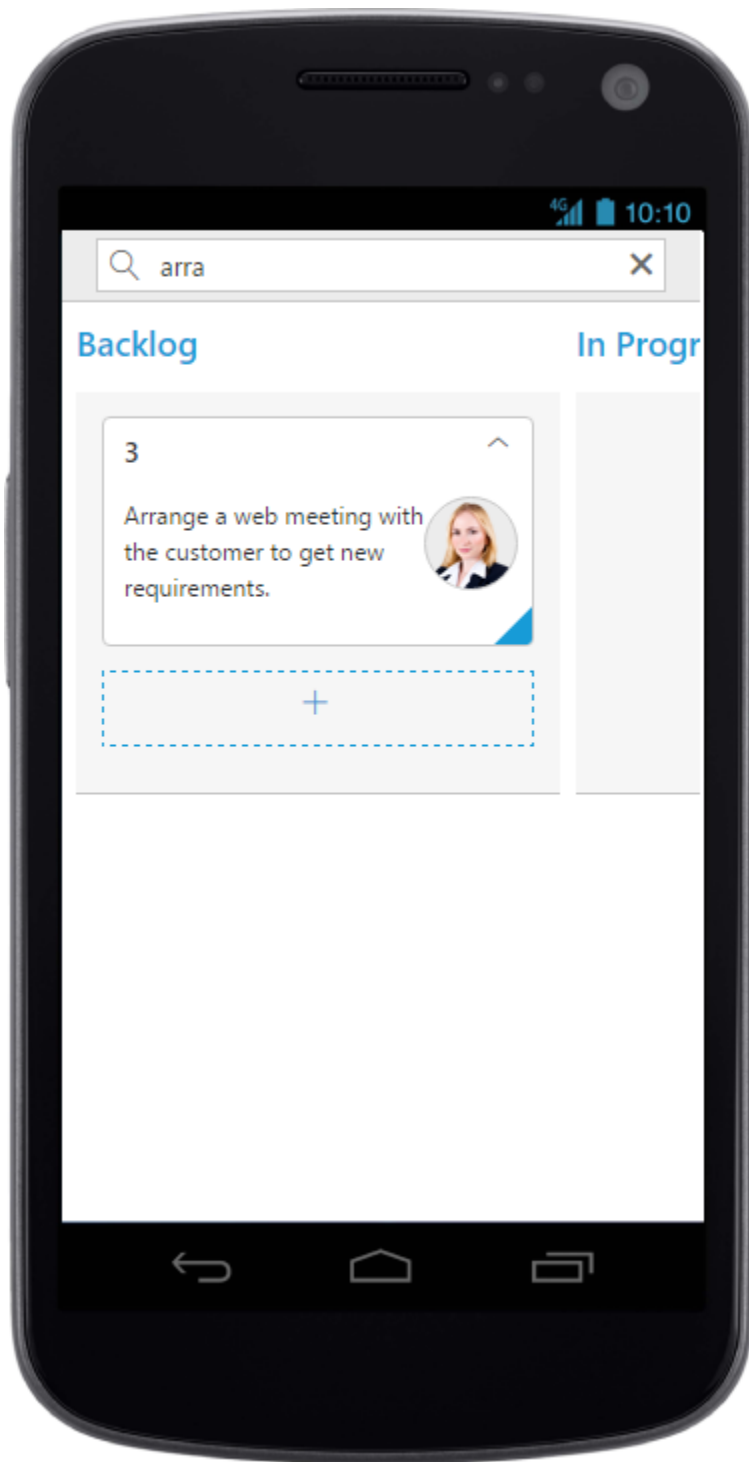
At the bottom of the form is a 'Save' button. The phone's navigation bar at the very bottom shows standard Android icons: back, home, and recent apps.

CRUD in mobile layout

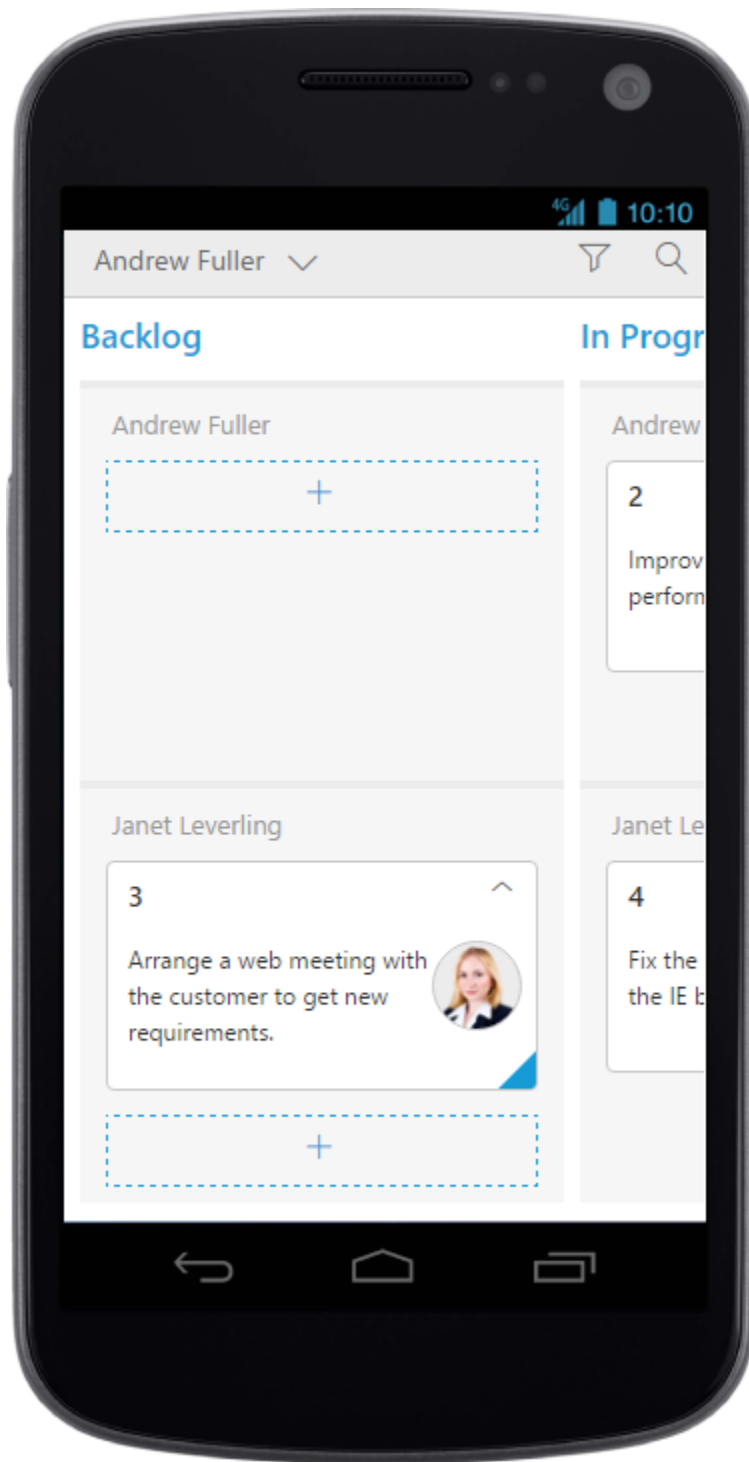


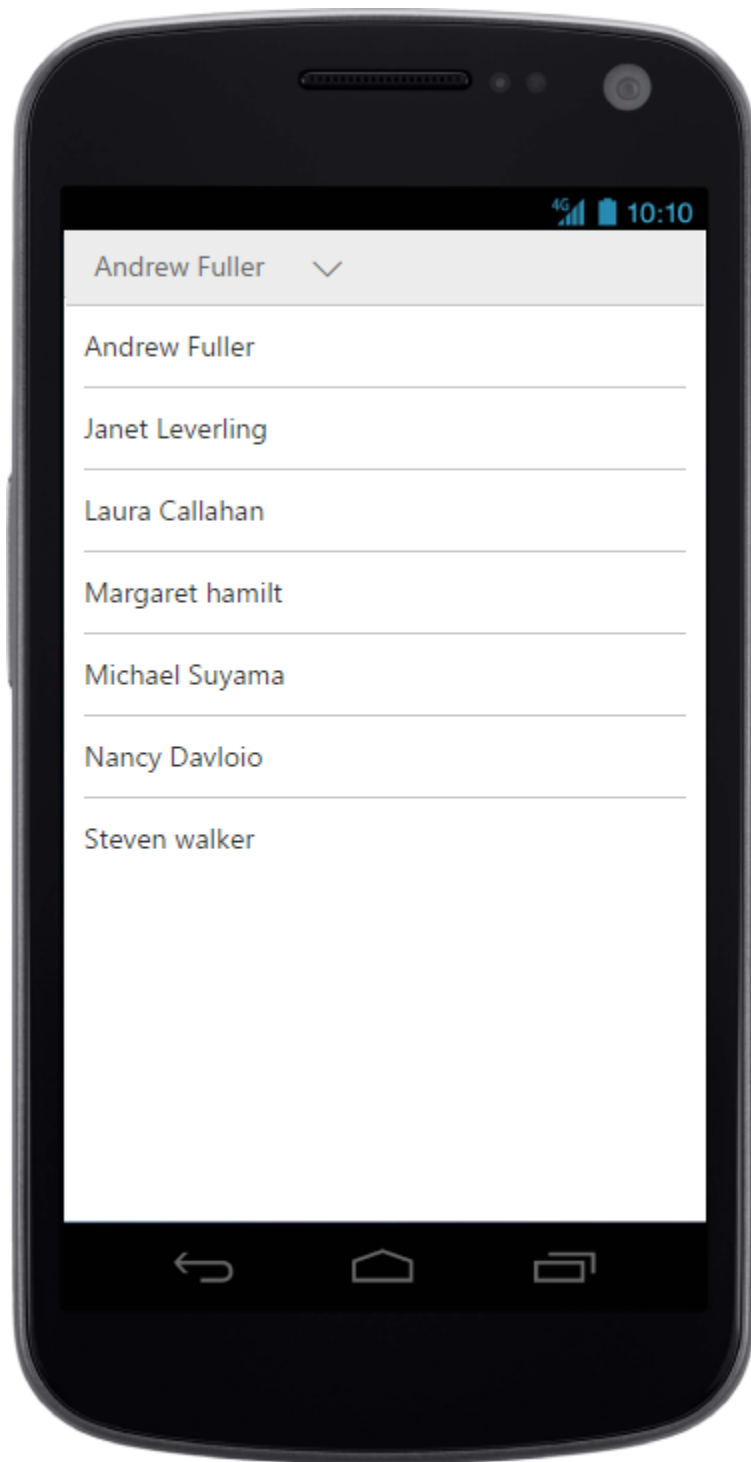
Filtering in mobile layout





Searching in mobile layout





Kanban with Swim-lane

#### Width

By default, the Kanban is adaptable to its parent container. It can adjust its width of columns based on parent container width. You can also assign width of [columns](#) in percentage.

The following code example describes the above behavior.

**HTML**

```
<div id='Kanban'></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
isResponsive: true,
columns: [
{ headerText: "Backlog", key: "Open",width:"10%" },
{ headerText: "In Progress", key: "InProgress", width: "10%" },
{ headerText: "Done", key: "Close", width: "10%" }
],
keyField: "Status",
fields: {
primaryKey: "Id",
content: "Summary",
tag: "Tags"
}
});
});
}
```

**Note:** allowScrolling should be false while defining width in percentage.

**Min Width**

Min Width is used to maintain minimum width for the Kanban. If the Kanban width is less than [minWidth](#) then the scrollbar will be displayed to maintain minimum width.

The following code example describes the above behavior.

**HTML**

```
<div id='Kanban'></div>
```

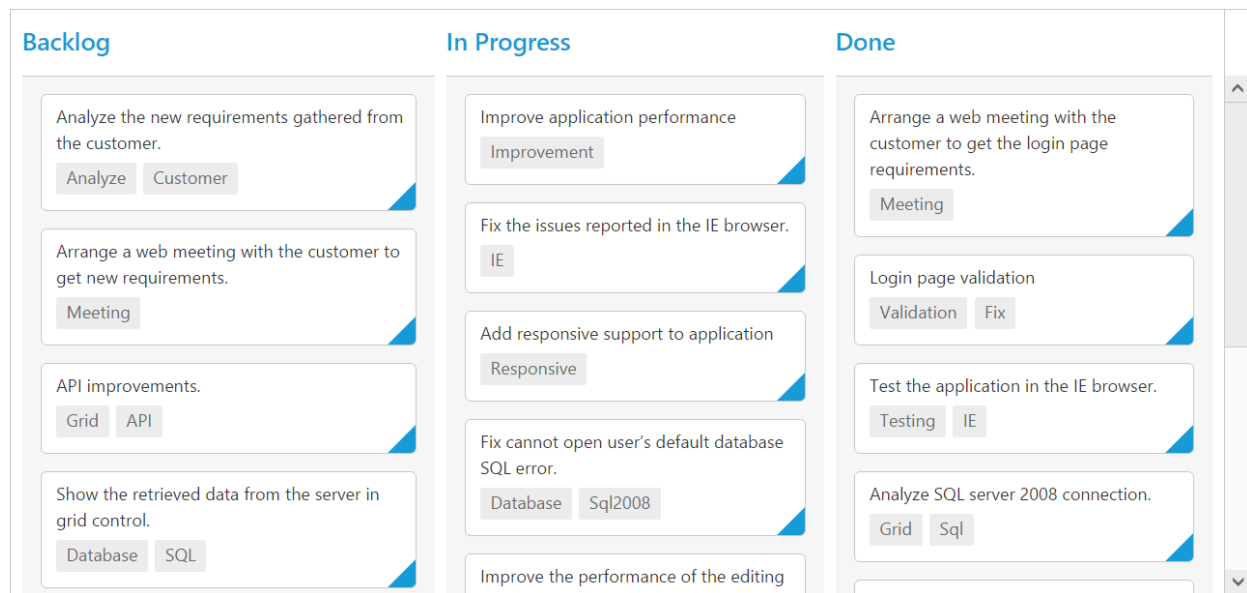
**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
minWidth: 700,
isResponsive: true,
columns: [
{ headerText: "Backlog", key: "Open", width: 120 },

```

```
{ headerText: "In Progress", key: "InProgress", width: 110 },
{ headerText: "Done", key: "Close", width: 110 }
],
keyField: "Status",
fields: {
primaryKey: "Id",
content: "Summary",
tag: "Tags"
}
});
});
}
```

The following output is displayed as a result of the above code example.



## Stacked Headers

The stacked headers helps you to group the logical columns in Kanban. It can be shown by setting `showStackedHeader` as true and by defining [stackedHeaderRows](#).

### Adding Stacked header columns

To stack columns in stacked header, you need to define [column](#) property in [stackedHeaderColumns](#) with field names of visible columns.

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

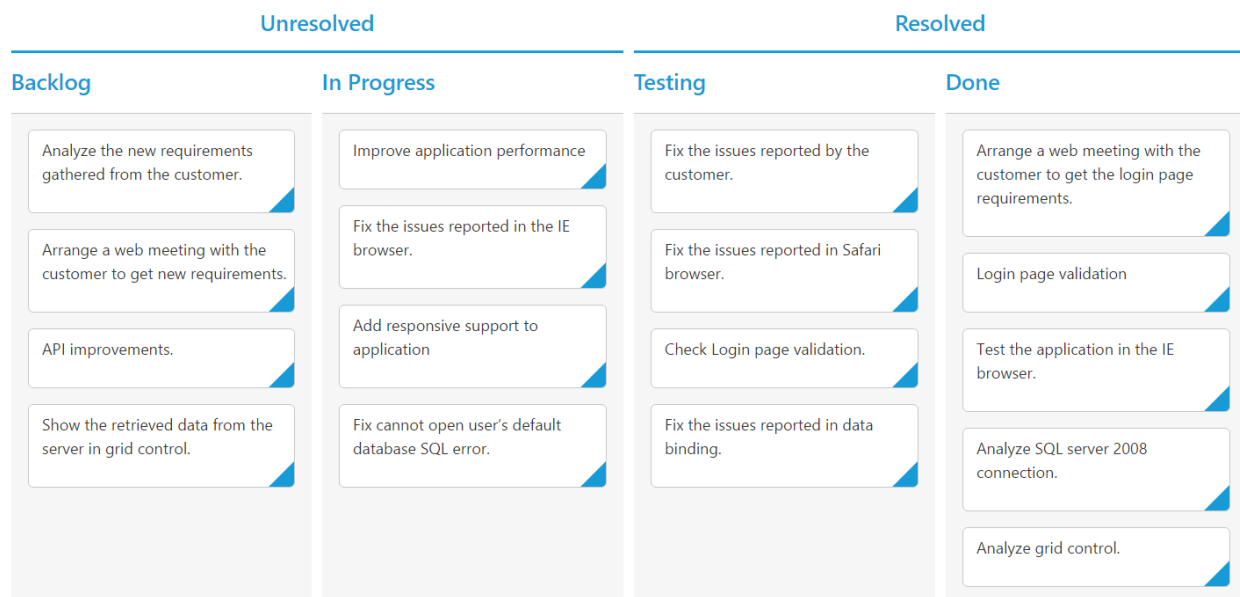
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
```

```

var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Testing", key: "Testing" }
],
keyField: "Status",
fields: {
primaryKey: "Id",
content: "Summary",
},
stackedHeaderRows: [{
stackedHeaderColumns: [{
headerText: "Unresolved",
column: "Backlog, In Progress"
}], {
headerText: "Resolved",
column: "Testing, Done"
}]
}
});

```

The following output is displayed as a result of the above code example.



## Context Menu

Context menu is used to improve user action with Kanban using popup menu. It can be shown by defining [contextMenuSettings.enable](#) as true. Context menu has option to add default items in [contextMenuSettings.menuItems](#) and customized items in [contextMenuSettings.customMenuItems](#).

**Note:** For using event handling in context menu , please refer this [API](#).

### Default Context Menu items

Please find the below table for default context menu items and its actions.

| Section | Context menu items | Action   |
|---------|--------------------|--|
| Header  | Hide Column        | Hide the current column                                    |
|         | Visible Columns    | Show the column if already hidden                          |
| Content | Add Card           | Start Add new card   |
| Card    | Edit Card          | Start Edit in current card                                 |
|         | Delete Card        | Delete the current card                                    |
|         | Top of Row         | Move the card to Top of Row                                |
|         | Bottom of Row      | Move the card to Bottom of Row                             |
|         | Move Up            | Move the card in Up direction                              |
|         | Move Down          | Move the card in Down direction                            |
|         | Move Left          | Move the card in Left direction                            |
|         | Move Right         | Move the card in Right direction                           |
|         | Move to Swimlane   | Move the card to Swim lane which is chosen from given list |
|         | Print Card         | Print the specific card                                    |

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
    $(function () {
        var data = new ej.DataManager(window.kanbanData).executeLocal(new
        ej.Query().take(20));
        var sample = new ej.Kanban($("#Kanban"), {
            dataSource: data,
            columns: [
                { headerText: "Backlog", key: "Open" },
                { headerText: "In Progress", key: "InProgress" },
                { headerText: "Testing", key: "Testing" }
            ],
            keyField: "Status",
            fields: {

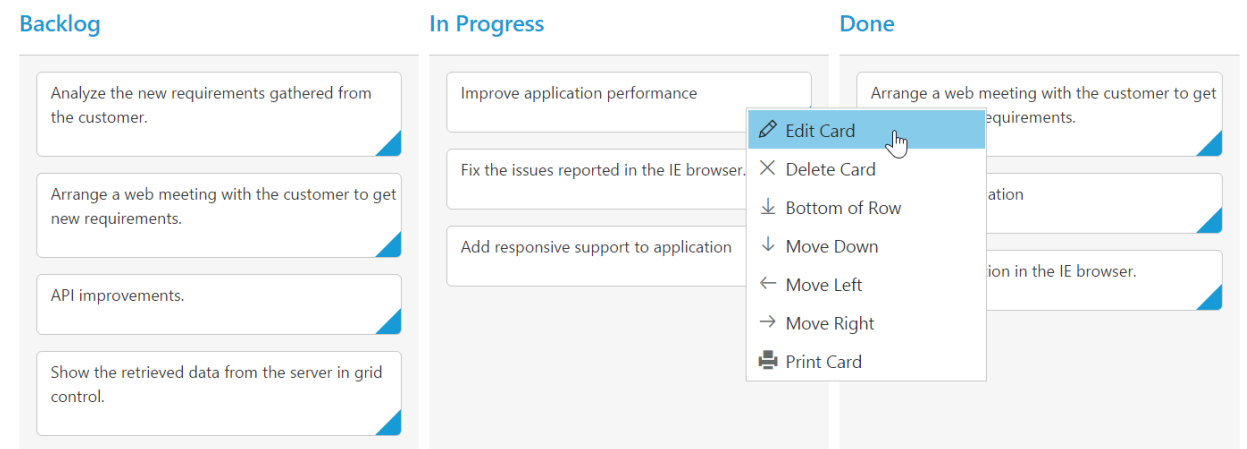
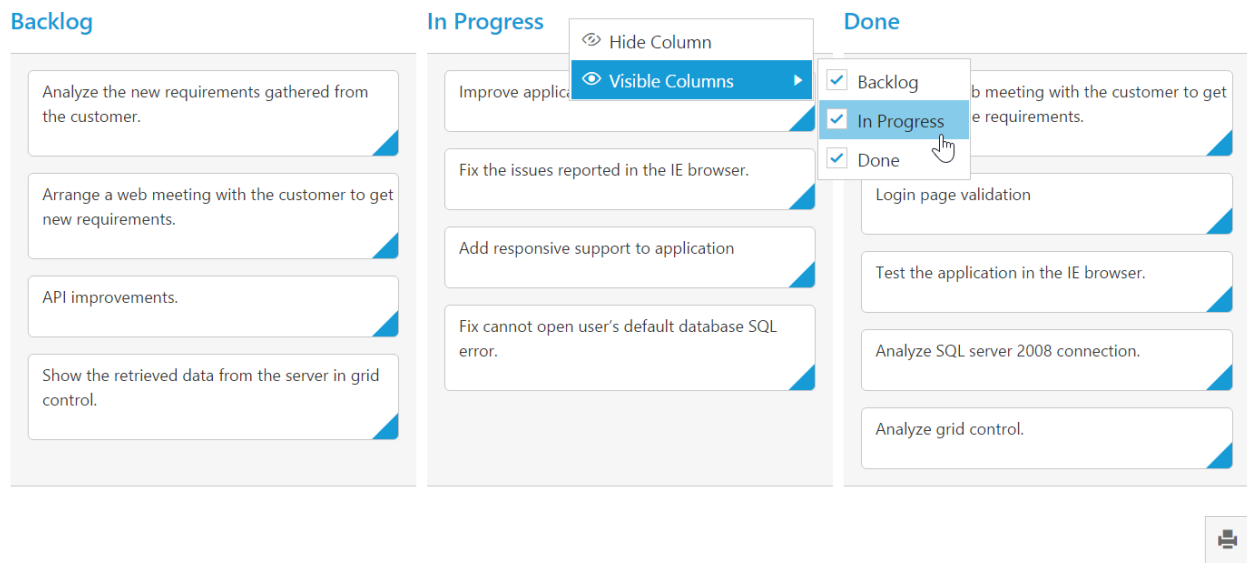
```

```

primaryKey: "Id",
content: "Summary",
},
contextMenuSettings: {
enable: true,
}
});
});
}

```

The following output is displayed as a result of the above code example.



### Custom Context Menu

Custom context menu is used to create your own menu item and its action. To add customized context menu items, you need to use [contextMenuSettings.customMenuItems](#) property and to bind required actions for this, use [contextClick](#) event.

The following code example describes the above behavior.

### HTML



```
<div id='Kanban'></div>
```

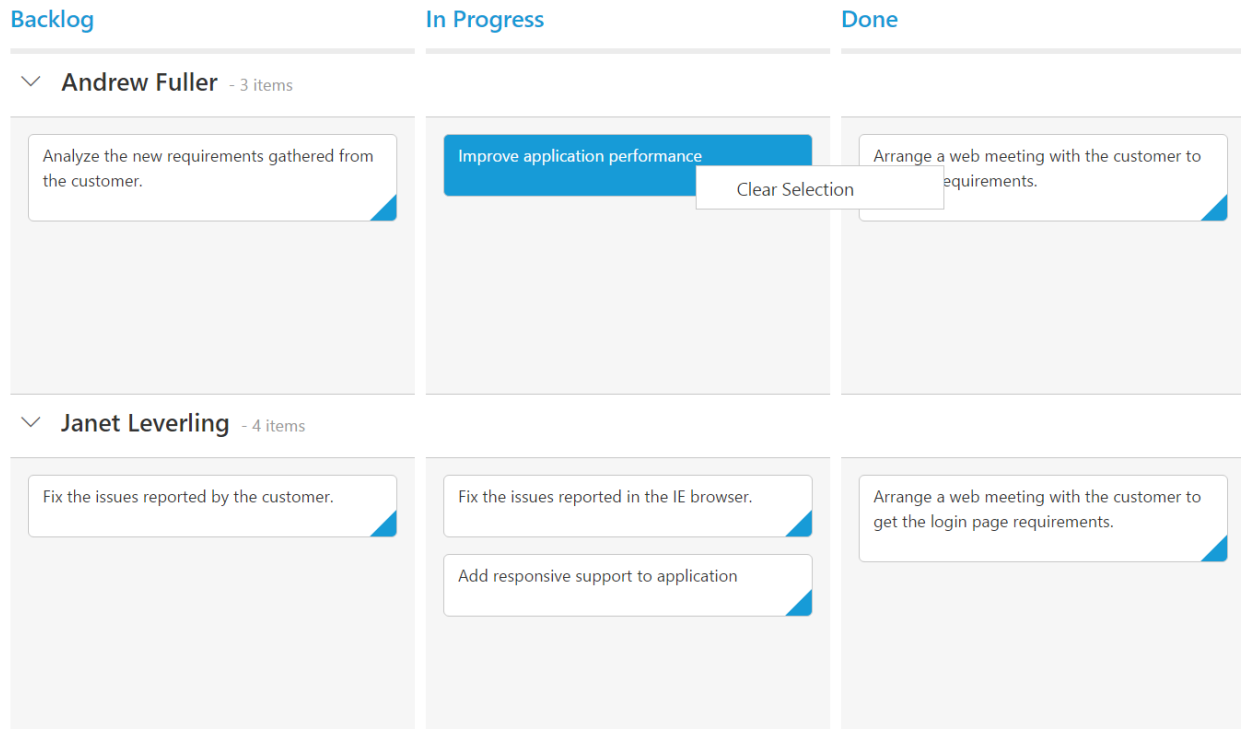
## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
contextClick: function (args) {
if (args.text == "Clear Selection")
this.KanbanSelection.clear();
},
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
primaryKey: "Id",
swimlaneKey: "Assignee",
content: "Summary",
tag: "Tags"
},
contextMenuSettings: {
enable: true,
menuItems: [],
customMenuItems: [{ text: "Clear Selection" }]
},
editSettings: {
editItems: [
{ field: "Id",
{ field: "Status", editType: ej.Kanban.EditingType.Dropdown },
{ field: "Assignee", editType: ej.Kanban.EditingType.Dropdown },
{ field: "Estimate", editType: ej.Kanban.EditingType.Numeric },
{ field: "Summary", editType: ej.Kanban.EditingType.TextArea }
],
allowEditing: true,
allowAdding: true
}
});
});
}

```

The following output is displayed as a result of the above code example.



### Sub Context Menu

Sub context menu is used to add customized sub menu to the custom context menu item. To add a sub context menu, you need to use [contextMenuSettings.subMenu](#) property and to bind required actions for this, use [contextClick](#) event.

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
contextClick: function (args) {
if (args.text == "Clear Selection")
this.KanbanSelection.clear();
else if (args.text != "Move to Column")
this.updateCard(args.data.id, args.data);
},
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
]
}
```

```

],
keyField: "Status",
fields: {
primaryKey: "Id",
swimlaneKey: "Assignee",
content: "Summary",
tag: "Tags"
},
contextMenuSettings: {
enable: true,
menuItems: [],
customMenuItems: [{ text: "Clear Selection" }, { text: "Move to Column",
template: "#submenu"}]
},
editSettings: {
editItems: [
{ field: "Id"},
{ field: "Status", editType: ej.Kanban.EditingType.Dropdown },
{ field: "Assignee", editType: ej.Kanban.EditingType.Dropdown },
{ field: "Estimate", editType: ej.Kanban.EditingType.Numeric },
{ field: "Summary", editType: ej.Kanban.EditingType.TextArea }
],
allowEditing: true,
allowAdding: true
}
});
});
}

```

The following output is displayed as a result of the above code example.

Backlog

In Progress

Done

Andrew Fuller - 3 items

Analyze the new requirements gathered from the customer.  
Analyze Customer

Improve application performance  
Improvement

Arrange a web meeting with the customer to get new requirements.  
Meeting

Clear Selection

Move to Column

Open

InProgress

Close

Janet Leverling - 4 items

Fix the issues reported by the customer.  
Customer

Fix the issues reported in the IE browser.  
IE

Add responsive support to application  
Responsive

Arrange a web meeting with the customer to get the login page requirements.  
Meeting

Copyright © 2001 - 2020 Syncfusion Inc.

1053

## Print

Set '[allowPrinting](#)' as true, to enable Print icon in the Kanban toolbar. You can use '[print\(\)](#)' method from Kanban instance to print the Kanban.

The following code example describes the above behavior.

## HTML

```
<div id='Kanban'></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), {
dataSource: data,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress" },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
allowTitle: true,
fields: {
primaryKey: "Id",
content: "Summary",
},
allowPrinting: true,
});
});
}
```

The following output is displayed as a result of the above code example.



## Backlog

1  
Analyze the new requirements gathered from the customer.

3  
Arrange a web meeting with the customer to get new requirements.

13  
API improvements.

15  
Show the retrieved data from the server in grid control.

## In Progress

2  
Improve application performance

4  
Fix the issues reported in the IE browser.

14  
Add responsive support to application

## Done

6  
Arrange a web meeting with the customer to get the login page requirements.

8  
Login page validation

10  
Test the application in the IE browser.

## Localization

## Localization

All text in Kanban can be localized using `ej.Kanban.Locale` object. Please find the table with list of properties and its value in locale object.

| Locale key words      | Text   |
|-----------------------|--|
| EmptyCard             | No cards to display  |
| SaveButton            | Save   |
| CancelButton          | Cancel   |
| EditFormTitle         | Details of   |
| AddFormTitle          | Add New Card   |
| SwimlaneCaptionFormat | "- {{:count}}{{if count == 1 }} item {{else}} items {{/if}}" |
| FilterSettings        | Filters:   |
| Min                   | Min  |
| Max                   | Max  |
| FilterOfText          | Of   |

|                |                  |
|----------------|------------------|
| Cards          | Cards            |
| ItemsCount     | Items Count :    |
| Unassigned     | Unassigned       |
| AddCard        | Add Card         |
| EditCard       | Edit Card        |
| DeleteCard     | Delete Card      |
| TopofRow       | Top of Row       |
| BottomofRow    | Bottom of Row    |
| MoveUp         | Move Up          |
| MoveDown       | Move Down        |
| MoveLeft       | Move Left        |
| MoveRight      | Move Right       |
| MovetoSwimlane | Move to Swimlane |
| HideColumn     | Hide Column      |
| VisibleColumns | Visible Columns  |
| PrintCard      | Print Card       |
| Search         | Search           |

The following code example describes the above behavior.

### HTML

```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
ej.Kanban.Locale["de-DE"] = {
  EmptyCard: "Keine Karten angezeigt werden",
  SaveButton: "Speichern",
  CancelButton: "stornieren",
  EditFormTitle: "Details von ",
  AddFormTitle: "Neue Karte hinzufügen",
  SwimlaneCaptionFormat: "- {{:count}}{{if count == 1 }} Artikel {{else}} Artikel {{/if}}",
  FilterSettings: "Filter:",
  FilterOfText: "Von",
  Max: "Max.",
  Min: "Min.",
  Cards: "Karten",
  ItemsCount: "Artikel Graf :",
}
```

```

Unassigned: "Nicht zugewiesen",
};
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
$(function () {
var data = new ej.DataManager(window.kanbanData).executeLocal(new
ej.Query().take(20));
var sample = new ej.Kanban($("#Kanban"), { dataSource: data,
locale: "de-DE",
enableTotalCount: true,
columns: [
{ headerText: "Backlog", key: "Open" },
{ headerText: "In Progress", key: "InProgress", constraints: { max: 2 } },
{ headerText: "Done", key: "Close" }
],
keyField: "Status",
fields: {
primaryKey: "Id",
swimlaneKey: "Assignee",
content: "Summary",
tag: "Tags"
}
});
});
}

```

The following output is displayed as a result of the above code example.

The Kanban board displays tasks organized by swimlanes and columns. The columns are Backlog, In Progress, and Done. The swimlanes are Andrew Fuller and Janet Leverling.

| Swimlane                    | Column      | Task Description  | Tags              |
|-----------------------------|-------------|---|-------------------|
| Andrew Fuller - 3 Artikel   | Backlog     | Analyze the new requirements gathered from the customer.                    | Analyze, Customer |
|                             | In Progress | Improve application performance   | Improvement       |
|                             | Done        | Arrange a web meeting with the customer to get new requirements.            | Meeting           |
| Janet Leverling - 4 Artikel | Backlog     | Fix the issues reported by the customer.                                    | Customer          |
|                             | In Progress | Fix the issues reported in the IE browser.                                  | IE                |
|                             | In Progress | Add responsive support to application                                       | Responsive        |
|                             | Done        | Arrange a web meeting with the customer to get the login page requirements. | Meeting           |

## Right to Left (RTL)

By default, Kanban render its text and layout from left to right. To customize Kanban's direction, you can change direction from LTR to RTL by using [enableRTL](#) as true.

The following code example describes the above behavior.

### HTML

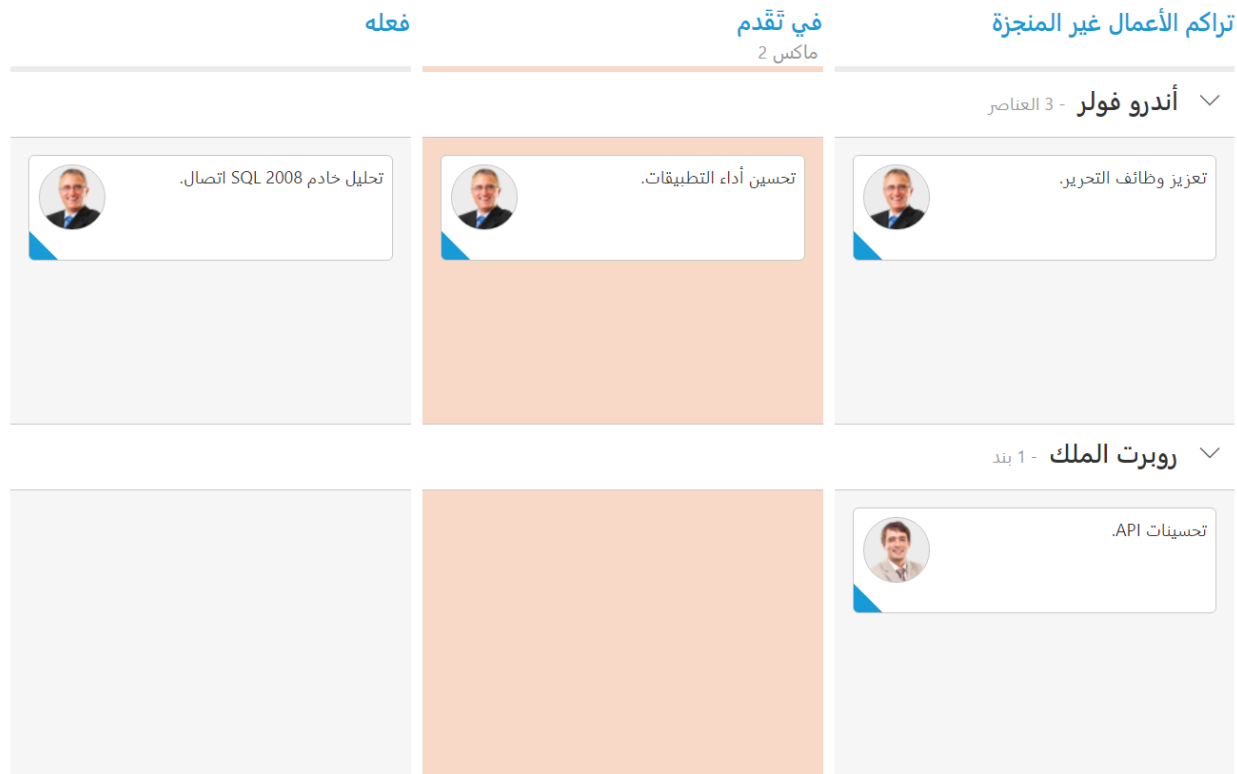
```
<div id='Kanban'></div>
```

### JAVASCRIPT

```
ej.Kanban.Locales["ar-AE"] = {
  EmptyCard: "لا بطاقات لعرض",
  SaveButton: "حفظ",
  CancelButton: "إلغاء",
  EditFormTitle: "تفاصيل",
  AddFormTitle: "إضافة بطاقة جديدة",
  SwimlaneCaptionFormat: "- {{:count}}{{if count == 1 }} بند {{else}} العناصر  
{{/if}}",
  FilterSettings: "مرشحات:",
  FilterOfText: "من",
  Max: "ماكس",
  Min: "دقيقة",
  Cards: "بطاقات",
  ItemsCount: "عدد العناصر:",
  Unassigned: "غير معين",
};
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module KanbanComponent {
  $(function () {
    var data = new ej.DataManager(window.kanbanData).executeLocal(new
    ej.Query().take(20));
    var sample = new ej.Kanban($("#Kanban"), {
      dataSource: data,
      enableRTL: true,
      locale: "ar-AE",
      columns: [
        { headerText: "تراكم الأعمال غير المنجزة", key: "Open" },
        { headerText: "في تقدم", key: "InProgress", constraints: { max: 2 } },
        { headerText: "فعله", key: "Close" }
      ],
      keyField: "Status",
      allowTitle: true,
      fields: {
        primaryKey: "Id",
        content: "Summary",
        imageUrl: "ImageUrl",
        swimlaneKey: "Assignee"
      }
    });
  });
}
```

The following output is displayed as a result of the above code example.





## Styling

### List of classes and its purposes

To modify Kanban appearance, you need to override default CSS of Kanban. Please find the list of CSS classes and its corresponding section in Kanban. Also you have an option to create your own custom theme for all JavaScript controls using our [Theme Studio](#)

| Section | CSS class       | Purpose of CSS class  |
|---------|-----------------|---|
| Root    | e-kanban        | This classes are in this root element (div) of Kanban control.  |
|         | e-js            |   |
| Header  | e-kanbantoolbar | This class is added at div element of Kanban toolbar.   |
|         | e-kanbanheader  | This is class is added in the root element of header element. In this class, You can override thin line between header and content of Kanban. |
|         | e-table         | This class is added at 'table' of Kanban header. This CSS class makes table width as 100 %.   |
|         | e-columnheader  | This class is added at 'tr' of Kanban header.   |
|         | e-headercell    | This class is added in 'th' element of Kanban header. You can override background color of header and border color                            |
|         | e-headercelldiv | This class is add in div which present 'th' element in header. You recommend you to use e-headercelldiv to override skeleton of header.       |

|      |                 |   |
|------|-----------------|---|
| Body | e-kanbancontent | This class is added at root of body content. This is to override background color of body.    |
|      | e-table         | This class is added to table of content. This CSS class makes table width as 100 %.           |
|      | e-swimlanerow   | This class is added to all swim lane â€™s in Kanban.  |
|      | e-columnrow     | This class is added to all next row of swimlane â€™s in Kanban                                |
|      | e-rowcell       | This class is added to all cells in Kanban. This is to override cells appearance and styling. |
|      | e-cardselection | This class is added to card div element of Kanban. This is override selection.                |
|      | e-hover         | This class adds to card of Kanban while hover cards.  |

## Web Accessibility

### Keyboard Navigation

Supported Keyboard Interactions keys with its description are tabulated as follows

| Interaction Keys | Description                   |
|------------------|-------------------------------|
| Alt + j          | Focus the Kanban              |
| Insert           | Insert card in Kanban         |
| Delete           | Delete card in Kanban         |
| F2               | Edit card in Kanban           |
| Enter            | Save edited or added          |
| Esc              | Cancel add or edit state      |
| Home             | Go to first card              |
| End              | Go to last card               |
| Up arrow         | Move to up card selection     |
| Down arrow       | Move to down card selection   |
| Right arrow      | Move to right card selection  |
| Left arrow       | Move to left card selection   |
| Ctrl + UpArrow   | Collapse All swim lane groups |
| Ctrl + DownArrow | Expand All swim lane groups   |
| Alt + UpArrow    | Collapse selected swim lane   |
| Alt + DownArrow  | Expand selected swim lane     |

|                    |                                |
|--------------------|--------------------------------|
| Alt + LeftArrow    | Collapse selected column       |
| Alt + RightArrow   | Expand selected column         |
| Shift + UpArrow    | Multi Selection by Up Arrow    |
| Shift + DownArrow  | Multi Selection by Down Arrow  |
| Shift + LeftArrow  | Multi Selection by Left Arrow  |
| Shift + RightArrow | Multi Selection by Right Arrow |

## LinearGauge

### Overview

The Gauge control for Essential Studio displays numerical information in the form of a scale that can be customized and oriented either vertically or horizontally. It comprises the following basic elements:

- Scales
- Pointers: bars and markers
- Ticks
- Labels
- Ranges

Three basic scale designs available are : rectangle, rounded rectangle, and thermometer.

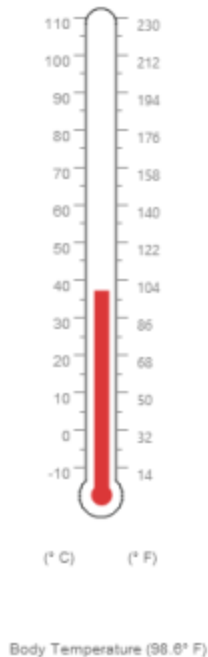
### Key Features

- **Interaction:** Allows you to directly interact with pointers on a gauge.
- **Indicators:** Indicate the state of a gauge either active or inactive.
- **Ranges:** Highlight the range of values on a gauge.
- **Pointers:** Allows you to add multiple pointers like bar pointers and marker pointers to a gauge.
- **Scale Direction:** Allows you to set scales direction either horizontally or vertically.
- **Animation:** Supports animation effects for pointers.
- **Custom Labels:** Allows you to add custom label text on any part of a gauge.
- **Scale Styles:** Three basic styles of scales are supported: rectangle, rounded rectangle, and thermometer.

### Getting Started

This section briefly explains on how to create a Linear Gauge control for your application.

- You can provide data for a Linear Gauge and display them in a required way. You can also customize the default Linear Gauge appearance to meet your requirements.
- In this example, you will learn how to create a Linear Gauge and how to design a thermometer, which can be used to check the body temperature of a person.



### Create a Linear Gauge

You can easily create the Digital Gauge widget by using the following steps.

1. First create an TypeScript Project and add the following references in the app.ts

For common getting started of TypeScript , you can refer [here](#).

The default type definition file **ej.web.all.d.ts** needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the **jquery.d.ts** file in your TypeScript application, which can be downloaded from [here](#).

2. Add the below script reference in the HTML page

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="https://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
```

```
</head>
<body>
</body>
</html>
```

In the above code, `ej.web.all.min.js` script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

3. Create a

tag.

### HTML

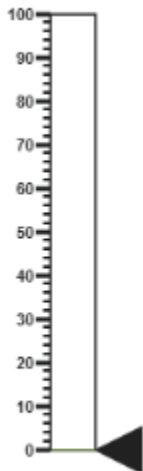
```
<html> <body> <div id="LinearGauge"></div> </body> </html>
```

4. Initialize the LinearGauge in ts file by using the `ej.LinearGauge` method.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var linearGaugeSample = new
    ej.datavisualization.LinearGauge($("#LinearGauge"));
  });
}
```

Run the above code example to get a default Linear Gauge with default values as follows.



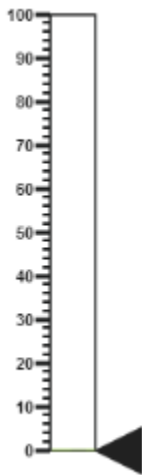
### Set Height and Width values

Basic attributes of each canvas elements are height and width. You can set the height and width of the gauge using the following code example. It sets the height and width of the canvas image where the thermometer is to be rendered.

#### JAVASCRIPT

```
$(function () {  
  var linearGaugeSample = new  
  ej.datavisualization.LinearGauge($("#LinearGauge"), {  
    height: 550,  
    width: 500,  
  });  
});
```

Run the above code example and you will get the following gauge as similar to default. Here height and width of the canvas are set for given values.



### Set Animation option and Label Color

- You can draw the Thermometer with some Label color to display the measurement value. For example give the labelColor as "#8c8c8c".
- Set the EnableAnimation property as false to avoid animation on the pointers.

#### JAVASCRIPT

```
$(function () {  
  var linearGaugeSample = new  
  ej.datavisualization.LinearGauge($("#LinearGauge"), {  
    height: 550,  
    width: 500,  
    labelColor: "#8c8c8c",  
    enableAnimation: false,  
  });  
});
```

Run the above code example and you will get the following gauge as the output.



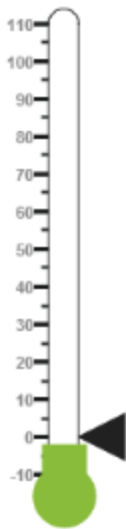
### Provide Scale Values

- The scale must have the appearance of a thermometer. By giving ScaleType as Thermometer, you can render a thermometer design.
- Minimum temperature can go up to -10 and maximum temperature can rise up to 110, so you can give minimum scale value as -10 and maximum value as 110.
- Set the location values such as vertical and horizontal position of the thermometer and give the thermometer height as Length.
- You can give the Minor Interval value as 5 to get the exact temperature on the patient.

### JAVASCRIPT

```
$(function () {  
  var linearGaugeSample = new  
  ej.datavisualization.LinearGauge($("#LinearGauge"), {  
    height: 550,  
    width: 500,  
    labelColor: "#8c8c8c",  
    enableAnimation: false,  
    scales: [{  
      type: "thermometer",  
      backgroundColor: "transparent",  
      minimum: -10,  
      maximum: 110,  
      minorIntervalValue: 5,  
      width: 20,  
      position: { x: 50, y: 18 },  
      length: 355,  
      border: { width: 0.5 }  
    }]  
  });  
});
```

Run the above code example and you will get the following gauge as the output.



### Add Pointers

In Linear Gauge the two types of pointers available are: Marker pointer and Bar pointer.

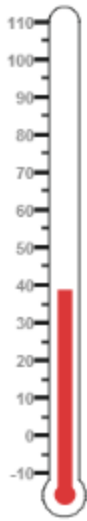
- Marker pointer is displayed as a pointer device that shows the actual values. But for your thermometer there is no need for the marker pointer. So you can hide the marker pointer by giving opacity as 0.
- Bar pointer acts as the mercury metal that shows the exact temperature of the patient. Set some of the basic properties of the Bar pointer such as Width, BarPointerDistanceFromScale, BarPointerValue and BarPointerBackgroundColor.

### JAVASCRIPT

```
$(function () {  
  var linearGaugeSample = new  
  ej.datavisualization.LinearGauge($("#LinearGauge"), {  
    height: 550,  
    width: 500,  
    labelColor: "#8c8c8c",  
    enableAnimation: false,  
    scales: [{  
      //Add the pointers customization code here  
      markerPointers: [{ opacity: 0 }],  
      barPointers: [{  
        width: 10,  
        distanceFromScale: -0.5,  
        value: 37,  
        backgroundColor: "#DB3738"  
      }],  
    }],  
  });  
});
```

On executing the above code sample renders a Linear Gauge with bar marker as follows.





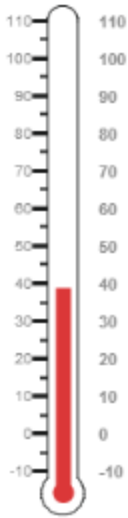
### Add Label Customization

- For thermometer, you can display the label value on two sides, to get temperature in different scales. For that you can add two label values in an array.
- To display the value around the scales, labels are used. You can customize the label placement, font (including its style and family) and its distance from scale.

### JAVASCRIPT

```
$(function () {  
    var linearGaugeSample = new  
    ej.datavisualization.LinearGauge($("#LinearGauge"), {  
        height: 550,  
        width: 500,  
        labelColor: "#8c8c8c",  
        enableAnimation: false,  
        scales: [{  
            labels: [{  
                placement: "near",  
                font: {  
                    size: "10px", fontFamily: "Segoe UI",  
                    fontStyle: "Normal"  
                }  
            }],  
        }],  
        {  
            placement: "far",  
            distanceFromScale: { x: 10 }  
        }],  
    });  
});
```

On executing the above code sample renders a customized Linear Gauge as follows.



### Add Tick Details

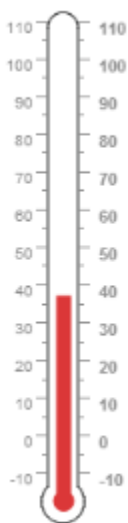
- Tick style has two values called major interval and minor interval. You can set major ticks width and height greater than Minor ticks. And you can give TickColor, for better visibility in light backgrounds.
- Here four tick details are used for both sides having minor and major ticks. To display the tick value add the following code example.

### JAVASCRIPT

```
$(function () {  
  var linearGaugeSample = new  
  ej.datavisualization.LinearGauge($("#LinearGauge"), {  
    height: 550,  
    width: 500,  
    labelColor: "#8c8c8c",  
    enableAnimation: false,  
    scales: [{  
      ticks: [{  
        type: "majorInterval",  
        height: 8,  
        width: 1,  
        color: "#8c8c8c",  
        distanceFromScale: { y: -4 }  
      }, {  
        type: "minorInterval",  
        height: 4,  
        width: 1,  
        color: "#8c8c8c",  
        distanceFromScale: { y: -4 }  
      }, {  
        type: "majorInterval",  
        placement: "far",  
        height: 8,  
        width: 1,  
        color: "#8c8c8c",
```

```
distanceFromScale: { y: -4 }
}, {
type: "minorInterval",
placement: "far",
height: 4,
width: 1,
color: "#8c8c8c",
distanceFromScale: { y: -4 }
}],
}]
});
});
```

On executing the above code sample renders a Linear Gauge with custom labels as follows.



### Add Custom Label Details

- Custom labels are used to specify the texts in the gauge.
- It can be customized through various properties.
- In order to show the custom labels, change the showIndicators property to true.
- Here you can use custom text to display three range descriptions.

### JAVASCRIPT

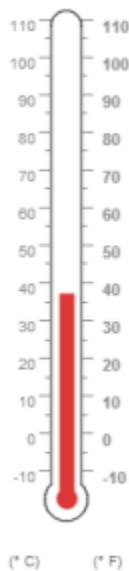
```
$(function () {
var linearGaugeSample = new
ej.datavisualization.LinearGauge($("#LinearGauge"), {
height: 550,
width: 500,
labelColor: "#8c8c8c",
enableAnimation: false,
scales: [{
showCustomLabels: true,
customLabels: [{
value: "(° C)",
```

```

position: { x: 44, y: 78 },
font: "Bold 12px Segoe UI", color: "#666666"
}, {
value: "(° F)",
position: { x: 56, y: 78 },
font: "Bold 12px Segoe UI", color: "#666666"
},
{
position: { x: 51, y: 90 },
font: "Bold 13px Segoe UI",
color: "#666666"
}]
}]
});
});

```

Run the above code example to get the following gauge as output.



### Change Scale from Degree to Fahrenheit

Add the function that converts the temperature in degree to Fahrenheit in the label, with an index value of 1.

#### JAVASCRIPT

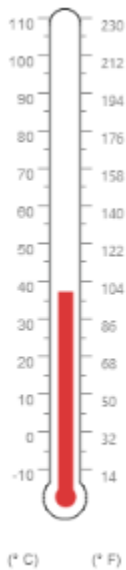
```

$(function () {
var linearGaugeSample = new
ej.datavisualization.LinearGauge($("#LinearGauge"), {
height: 550,
width: 500,
labelColor: "#8c8c8c",
enableAnimation: false,
drawLabels: function () {
var args = $("#LinearGauge").data("ejLinearGauge");
if (args.label.index == 1) {

```

```
args.style.textValue = (args.label.value * (9 / 5)) + 32;
args.style.font = "Normal 10px Segoe UI";
}
}
});
});
```

Run the above code example and you will get the following gauge as output.



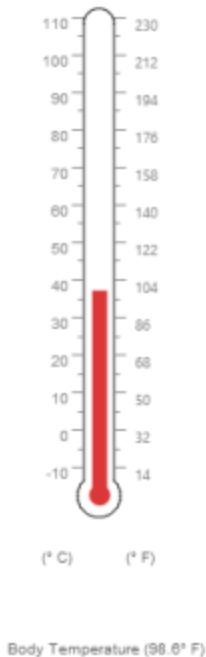
### Add Custom Label for Current Value

Add the function that displays the current temperature value in the custom label.

#### JAVASCRIPT

```
$(function () {
var linearGaugeSample = new
ej.datavisualization.LinearGauge($("#LinearGauge"), {
height: 550,
width: 500,
labelColor: "#8c8c8c",
enableAnimation: false,
drawLabels: function () {
var args = $("#LinearGauge").data("ejLinearGauge");
if (args.label.index == 1) {
args.style.textValue = (args.label.value * (9 / 5)) + 32;
args.style.font = "Normal 10px Segoe UI";
}
},
drawCustomLabels : function () {
var args = $("#LinearGauge").data("ejLinearGauge");
if (args.customLabelIndex == 2) {
var temp = args.scaleElement.barPointers[0].value;
var faValue = (temp * (9 / 5)) + 32;
if (temp == -10) {
```

```
args.style.textValue = "Very Cold Weather" + "(" + faValue.toFixed(1) + "° F)";  
}  
else if ((temp > -10 && temp < 0) || (temp > 0 && temp < 15)) {  
args.style.textValue = "Cool Weather" + "(" + faValue.toFixed(1) + "° F)";  
}  
else if (temp == 0) {  
args.style.textValue = "Freezing point of Water" + "(" + faValue.toFixed(1) + "° F)";  
}  
else if (temp >= 15 && temp < 30) {  
args.style.textValue = "Room Temperature" + "(" + faValue.toFixed(1) + "° F)";  
}  
else if (temp == 30) {  
args.style.textValue = "Beach Weather" + "(" + faValue.toFixed(1) + "° F)";  
}  
else if (temp == 37) {  
args.style.textValue = "Body Temperature" + "(" + faValue.toFixed(1) + "° F)";  
}  
else if (temp == 40) {  
args.style.textValue = "Hot Bath Temperature" + "(" + faValue.toFixed(1) + "° F)";  
}  
else if (temp > 40 && temp < 100) {  
args.style.textValue = "Very Hot Temperature" + "(" + faValue.toFixed(1) + "° F)";  
}  
else if (temp == 100) {  
args.style.textValue = "Boiling point of Water" + "(" + faValue.toFixed(1) + "° F)";  
}  
}  
});  
});
```



## Basic Settings

### Adding Dimension

- The basic customization for any control is setting the dimension. Here dimension refers the two major attributes, **height** and **width**. The height and width assigned in the control will render the canvas element in the given size.
- The value attribute is used to set all pointer value in the Linear Gauge control. The attributes, **minimum** and **maximum** value are used to set the minimum value and maximum value for all the scales exist in the Linear Gauge control.

### HTML

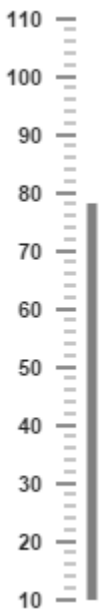
```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../../tsfiles/ej.web.all.d.ts"></reference>
module LinearGaugeComponent {
    $(function () {
        // For Linear Gauge rendering
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            // For setting linear gauge width
            width: 300,
            // For setting linear gauge height
            height: 500,
            // For setting linear gauge minimum value
            minimum: 10,
            // For setting linear gauge maximum value
            maximum: 110,
```

```
// For setting linear gauge pointer value
value: 78,
// Adding scale collection
scales: [{
border: { color: "transparent", width: 0 },
showMarkerPointers: false, showBarPointers: true,
// Adding bar pointer collection
barPointers: [{ width: 5, backgroundColor: "Grey" }],
// Adding ticks collection
ticks: [{
type: "MajorInterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
},
{
type: "MinorInterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
}]
}];
});
}
```

Execute the above code to render the following output.



#### Adding frame

- Frame is the element that decides the appearance of the **Linear Gauge**. You can customize it by using the object called **frame**. It contains frame inner width, frame outer width and frame background image URL properties.
- The **innerWidth** of the frame defines the distance between the canvas element and the frame and the **outerWidth** refers to distance from the frame. **backgroundImageUrl** is used to set the background image for the frame.



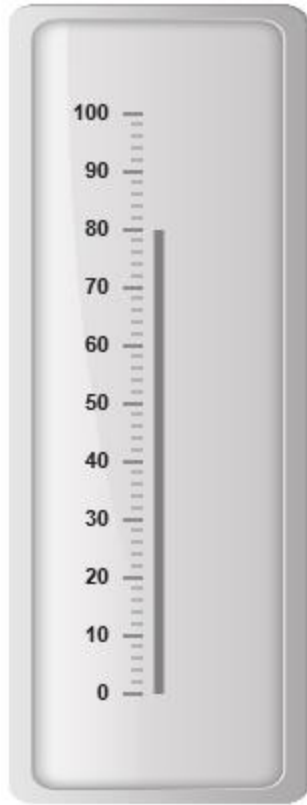
## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
$(function () {  
    // For Linear Gauge rendering  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        value: 80,  
        frame: {  
            // For setting frame inner width  
            innerWidth: 8,  
            // For setting frame outer width  
            outerWidth: 10,  
            // For setting back ground Image URL  
            backgroundImageUrl: "../images/gauge/Gauge_linear_light.png"  
        },  
        // Adding scale collection  
        scales: [{  
            backgroundColor: "transparent",  
            border: { color: "transparent", width: 0 },  
            showMarkerPointers: false, showBarPointers: true,  
            // Adding bar pointer collection  
            barPointers: [{ width: 5, backgroundColor: "Grey" }],  
            // Adding ticks collection  
            ticks: [{  
                type: "MajorInterval", width: 2,  
                color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }  
            },  
            {  
                type: "MinorInterval", width: 1, height: 6,  
                color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }  
            }  
        ]  
        }  
    ]  
});
```

Execute the above code to render the following output.



### Appearance

- The attribute **orientation** is used to render the Linear Gauges either in horizontal position or vertical position. You can set the background color for the Linear Gauge for better appearance using the **backgroundColor** property. **borderColor** specifies the **borderColor** of the Linear Gauge. You can also add gradient effects to Linear Gauge with the help of **pointerGradient1** and **pointerGradient2** attribute.
- Theme is the basic property of any control. It is used to set the **theme** for Linear Gauge. There are two types of themes used for Linear Gauge such as
  - FlatLight
  - FlatDark

### HTML

```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

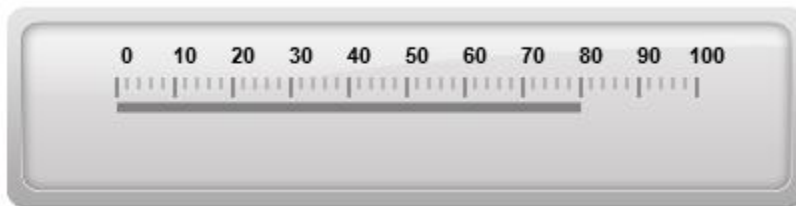
```
$(function () {  
  // For Linear Gauge rendering  
  var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
    enableAnimation: false,  
    width: 400,  
    height: 100,  
    value: 80,  
    // For setting theme
```

```

theme: "FlatLight",
// For setting Orientation
orientation: "Horizontal",
// For setting label color
labelColor: "Black",
//For Adding Frame
frame: {
backgroundImageUrl: "../images/gauge/Gauge_linear_light1.png"
},
//For Adding Scales
scales: [{
background-color: "transparent",
direction: ej.datavisualization.LinearGauge.Directions.Clockwise,
border: { color: "transparent", width: 0 },
showMarkerPointers: false, showBarPointers: true,
//For Adding bar pointers
barPointers: [{ width: 5, backgroundColor: "Grey" }],
//For Adding label pointers
labels: [{ angle: 90, distanceFromScale: { x: 5, y: -5 } }],
//For Adding ticks
ticks: [{
type: "MajorInterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 0, y: 0 }
},
{
type: "MinorInterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 0, y: 0 }
}]
}
});

```

Execute the above code to render the following output.



### Responsive

- For any display devices, the control is to be rendered based on the space in that device. The control must be responsive. For this purpose resizing property is present in **Linear Gauge** control.
- The **Linear Gauge** renders with the specified value. When the browser changes its size, the canvas element checks the dimension with its parent element and if there are any changes in parent dimension, gauge control also changes the dimension based on its parent changes. You can enable this feature using `isResponsive` property.

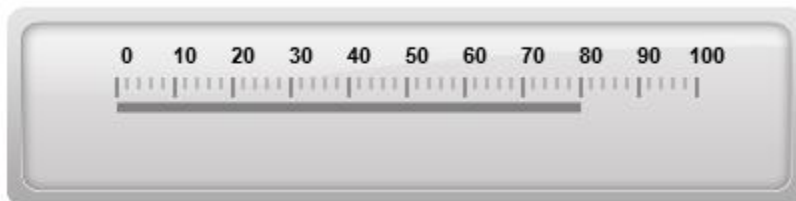
### HTML

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
$(function () {
    // For Linear Gauge rendering
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
        enableAnimation: false,
        width: 400,
        height: 100,
        value: 80,
        orientation: "Horizontal",
        labelColor: "Black",
        //For enabling responsible layout
        isResponsive: true,
        //For Adding Frame
        frame: {
            backgroundImageUrl: "../images/gauge/Gauge_linear_light1.png"
        },
        //For Adding Scale
        scales: [{
            backgroundColor: "transparent",
            direction: ej.datavisualization.LinearGauge.Directions.Clockwise,
            border: { color: "transparent", width: 0 },
            showMarkerPointers: false, showBarPointers: true,
            //For Adding bar pointer collection
            barPointers: [{ width: 5, backgroundColor: "Grey" }],
            //For Adding label collection
            labels: [{ angle: 90, distanceFromScale: { x: 5, y: -5 } }],
            //For Adding tick collection
            ticks: [{
                type: "MajorInterval", width: 2,
                color: "#8c8c8c", distanceFromScale: { x: 0, y: 0 }
            },
            {
                type: "MinorInterval", width: 1, height: 6,
                color: "#8c8c8c", distanceFromScale: { x: 0, y: 0 }
            }
        ]
    }
    });
});
```

Execute the above code to render the following output.



Responsiveness of the linear gauge is controlled by using `enableResize` property.

**JAVASCRIPT**

```
<div id="LinearGauge1"></div>
```

```
<script>
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
enableResize: true });
</script>
```

### Localization

**LinearGauge** supports localization for its axis labels and tooltip. To render the gauge with specific culture you have to refer the corresponding globalize culture script and need to specify the culture name in **locale** property of gauge.

**Enable Group Separator** is used to Convert the date object to string while using the locale settings, you can set **enableGroupSeparator** property as **true**.

### JAVASCRIPT

```
<div id="LinearGauge1"></div>
<script>
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
locale : "en-US" ,
enableGroupSeparator: true
});
</script>
```

### Interaction and Animation

- **Linear Gauge** control contains **Interaction** feature. You can use this interaction feature to change the pointer values manually either by clicking or dragging the pointer over the **Gauge**. It dynamically changes the value of pointer when dragged. To Enable/Disable the user interaction you can use the **readOnly** Boolean property. The user interaction option is enabled when you set **readOnly** property as false. By default it holds the true value.
- **Linear Gauge** contains another attractive concept called **Animation**. The animation option enables the movement of the pointer from the minimum value to the current value. You can use animation option to change the pointer value dynamically. You can enable/ disable it using **enableAnimation** property. To enable animation set **enableAnimation** to "true".
- By default it holds the true value. You can control the speed of the pointer during animating using **animationSpeed**. It is a numerical value that holds the time in milliseconds. That is when setting value is 1000, it is considered as 1 second.

### HTML

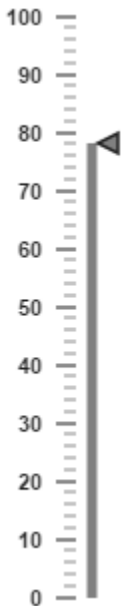
```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module LinearGaugeComponent {
$(function () {
// For Linear Gauge rendering
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
//For enable Animation
```

```
enableAnimation: true,  
//For enable Animation speed  
animationSpeed: 1000,  
//For enable interaction  
readOnly: false,  
value: 78,  
//For Adding scale collection  
scales: [{  
border: { color: "transparent", width: 0 },  
showBarPointers: true,  
//For Adding bar pointer collection  
barPointers: [{ width: 5, backgroundColor: "Grey" }],  
//For Adding marker pointer collection  
markerPointers: [{  
width: 10, length: 10, backgroundColor: "Grey", distanceFromScale: -12  
}],  
//For Adding ticks collection  
ticks: [{  
type: "majorinterval", width: 2,  
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }  
},  
{  
type: "minorinterval", width: 1, height: 6,  
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }  
}]  
}];  
});  
});  
}
```

Execute the above code to render the following output.



#### Enable Marker Pointer Animation

Specifies the animate state for marker pointer, you can set `enableMarkerPointer` property as **true**

**JAVASCRIPT**

```
<div id="LinearGauge1"></div>
<script>
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
enableMarkerPointerAnimation: true, value:50});
</script>
```

**Scales**

**Scales** are the basic functional block of the **Linear Gauge**. You can improve the appearance of scales by customizing it. The functional blocks of **Linear Gauge** are

- Marker Pointers
- Bar Pointer
- Labels
- Custom Labels
- Indicators
- Ticks
- Ranges

**Adding scale collection**

**Scale** is the basic element of **Linear Gauge**. Scale collection is directly added to the gauge object. Refer the following code example to add scale collection in **Gauge** control.

**HTML**

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module LinearGaugeComponent {
$(function () {
// For Linear Gauge rendering
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
enableAnimation: false,
frame: { backgroundImageUrl: "../../images/gauge/Gauge_linear_light.png" },
// For Adding Scale collection
scales: [{
width: 8,
position: { x: 20, y: 50 },
backgroundColor: "Grey",
border: { color: "Grey", width: 1 },
showMarkerPointers: true, showBarPointers: false,
// For Adding label collection
labels: [{ distanceFromScale: { x: 50, y: 0 } }],
// For Adding marker pointer collection
markerPointers: [{
type: "pentagon", placement: "near",
length: 10, width: 20, distanceFromScale: 20, backgroundColor: "#FE8282"
}],
// For Adding tick collection
```

```
ticks: [{
  type: "majorinterval", width: 2,
  color: "#8c8c8c", distanceFromScale: { x: 30, y: 0 }
},
{
  type: "minorinterval", width: 1, height: 6,
  color: "#8c8c8c", distanceFromScale: { x: 30, y: 0 }
}]
});
});
}
```

Execute the above code to render the following output.



### Scale Customization

#### Colors and Border

- The **Scale** border is modified with **border** object. It has two border property, **color** and **width** are used to customize the border color of the scale and border width of the scale. Setting the background color improves the look and feel of the **Linear Gauge**. You can customize the background color of the scale using **backgroundColor**.
- Scales are used to enable or disable various properties such as **showRanges**, **showIndicators**, **showCustomLabels**, **showLabels**, **showTicks**, **showBarPointers** and **showMarkerPointers**. Enable/disable is done by setting the property into two states either **"true"** or **"false"**. You can adjust the Opacity of the scale with **opacity** property.



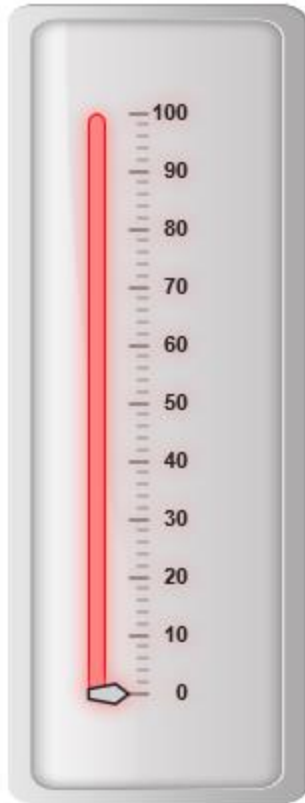
## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
$(function () {  
    //For Linear gauge rendering  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        enableAnimation: false,  
        frame: { backgroundImageUrl: "../images/gauge/Gauge_linear_light.png"},  
        //For Adding scale collection  
        scales: [{  
            width: 8,  
            position: { x: 20, y: 50 },  
            backgroundColor: "#FE8282",  
            border: { color: "Red", width: 1 },  
            //For Adding opacity  
            opacity: 0.5,  
            //For Adding Shadow offset  
            shadowOffset: 10,  
            type: "roundedrectangle",  
            showMarkerPointers: true,  
            showBarPointers: false,  
            //For Adding label collection  
            labels: [{ distanceFromScale: { x: 50, y: 0 } }],  
            //For Adding marker pointer collection  
            markerPointers: [{  
                type: "pentagon", placement: "near", length: 10,  
                width: 20, distanceFromScale: 20, backgroundColor: "#C9E1E5"  
            }],  
            //For Adding ticks collection  
            ticks: [{  
                type: "majorinterval", width: 2,  
                color: "#8c8c8c", distanceFromScale: { x: 30, y: 0 }  
            },  
            {  
                type: "minorinterval", width: 1, height: 6,  
                color: "#8c8c8c", distanceFromScale: { x: 30, y: 0 }  
            }  
        ]  
    }];  
});
```

Execute the above code to render the following output.



### Appearance

- You can improve the appearance of **Linear Gauge** using various properties. You can set the interval values for the scale with `majorIntervalValue` and `minorIntervalValue` properties and maximum and minimum value by `minimum` and `maximum` property. The `width` property is used to set the scale bar width.
- You can also adjust the Opacity of the scale with `opacity` property. The value for opacity lies between 0 and 1. **Linear Gauge** contains two scale directions, clockwise and counter clockwise. It can be set with `direction` property.

### HTML

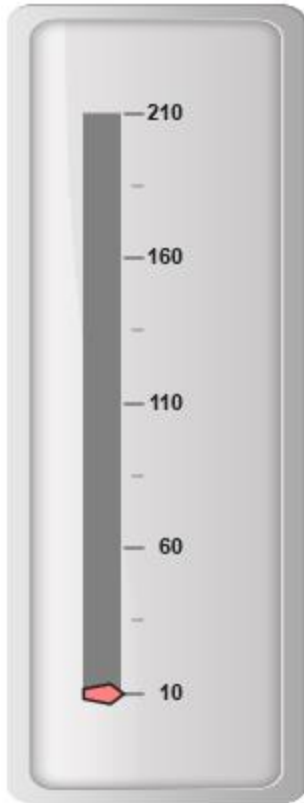
```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    // For Linear Gauge rendering  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        enableAnimation: false,  
        frame: { backgroundImageUrl: "../images/gauge/Gauge_linear_light.png" },  
        scales: [{  
            //For Adding scale width  
            width: 18,  
            //For Adding scale minimum value  
            minimum: 10,  
        }],  
    });  
});
```

```
//For Adding scale maximum value
maximum:210,
//For Adding scale minor interval value
minorIntervalValue:25,
//For Adding scale major interval value
majorIntervalValue:50,
//For Adding scale direction
direction: "counterclockwise",
position: { x: 20, y: 50 },
backgroundColor: "Grey",
border: { color: "Grey", width: 1 },
showMarkerPointers: true, showBarPointers: false,
//Adding label collection
labels: [{ distanceFromScale: { x: 50, y: 0 } }],
//Adding marker pointer collection
markerPointers: [{
  type: "pentagon", placement: "near",
  length: 10, width: 20, distanceFromScale: 20,
  backgroundColor: "#FE8282"
}],
//Adding tick collection
ticks: [{
  type: "majorinterval", width: 2,
  color: "#8c8c8c", distanceFromScale: { x: 30, y: 0 }
},
{
  type: "minorinterval", width: 1, height: 6,
  color: "#8c8c8c", distanceFromScale: { x: 30, y: 0 }
}]
});
});
```

Execute the above code to render the following output.



### Scale Types

Scale Type is an element which decides the appearance of the gauge. **Linear Gauge** contains three scale type such as,

- Rectangle
- Rounded Rectangle
- Thermometer

### Rectangle

For rectangular scale type, the scale renders with rectangular structure. Refer the following code example.

#### HTML

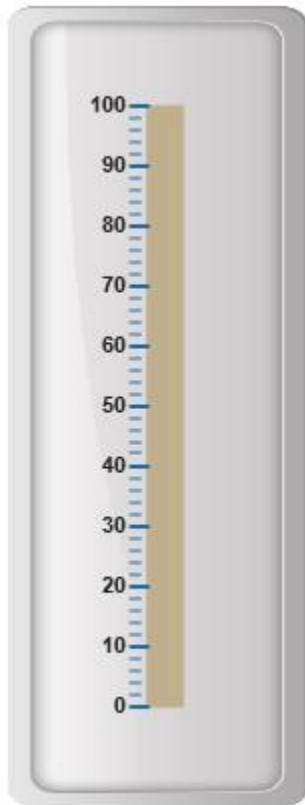
```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {  
  // For Linear Gauge rendering  
  var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
    enableAnimation: false,  
    frame: { backgroundImageUrl: "../images/gauge/Gauge_linear_light.png" },  
    //For Adding Scale collection  
    scales: [{  
      width: 18, length: 300,
```

```
position: { x: 54, y: 50 },
//For Adding Scale type as rectangle
type: "rectangle",
backgroundColor: "#C0B08E",
border: { color: "#C0B08E", width: 1 },
showMarkerPointers: false, showBarPointers: false,
//For Adding tick collection
ticks: [{
  type: "majorinterval", width: 2,
  color: "#206BA4", distanceFromScale: { x: -27, y: 0 },
  placement: "far"
},
{
  type: "minorinterval", width: 1, height: 6,
  color: "#206BA4", distanceFromScale: { x: -27, y: 0 },
  placement: "far"
}]
}];
});
```

Execute the above code to render the following output.



### Rounded Rectangle

For rounded rectangular scale `type`, the scale renders as rectangular structure but with constant radius rounded corner. Refer the following code example.

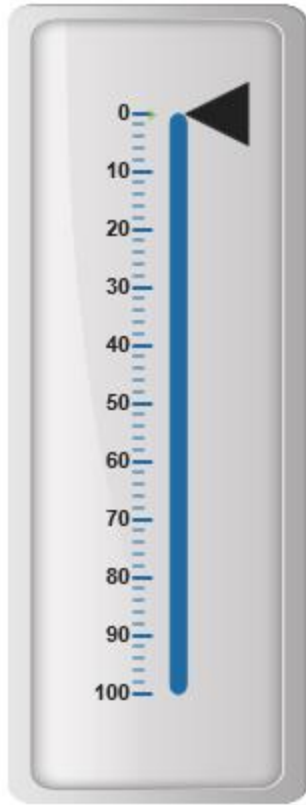
## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
$(function () {  
  // For Linear Gauge rendering  
  var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
    enableAnimation: false,  
    frame: { backgroundImageUrl: "../images/gauge/Gauge_linear_light.png" },  
    //For Adding Scales  
    scales: [{  
      width: 8,  
      direction: ej.datavisualization.LinearGauge.Directions.Clockwise,  
      position: { x: 60, y: 50 },  
      //Adding scale type as rounded rectangle  
      type: "roundedrectangle",  
      backgroundColor: "#206BA4",  
      border: { color: "#206BA4", width: 1 },  
      //Adding label collection  
      labels: [{ distanceFromScale: {x:-20,y:0}}],  
      //Adding tick collection  
      ticks: [{  
        type: "majorinterval", width: 2,  
        color: "#206BA4", distanceFromScale: { x: -27, y: 0 },  
        placement: "far"  
      },  
      {  
        type: "minorinterval", width: 1, height: 6,  
        color: "#206BA4", distanceFromScale: { x: -27, y: 0 },  
        placement: "far"  
      }  
    ]  
  }  
});
```

Execute the above code to render the following output.



### Thermometer

For thermometer scale **type**, the scale renders as thermometer structure with rounded bottom. Refer the following code example.

#### HTML

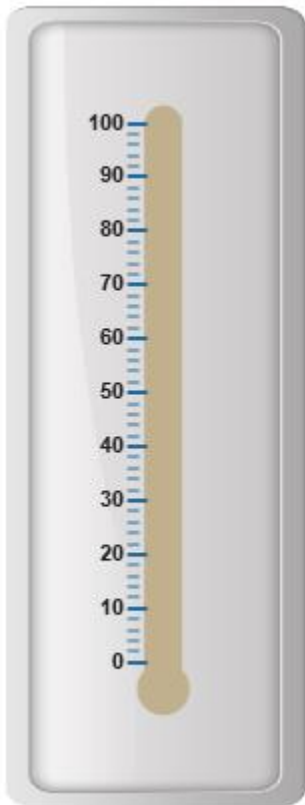
```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {  
    // For Linear Gauge rendering  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        enableAnimation: false,  
        frame: { backgroundImageUrl: "../images/gauge/Gauge_linear_light.png" },  
        scales: [{  
            width: 18, length: 300,  
            position: { x: 54, y: 50 },  
            //Adding scale type as thermometer  
            type: "thermometer",  
            backgroundColor: "#C0B08E",  
            border: { color: "#C0B08E", width: 1 },  
            showMarkerPointers: false, showBarPointers: false,  
            //Adding tick collection  
            ticks: [{  
                type: "majorinterval", width: 2,  
                color: "#206BA4", distanceFromScale: { x: -27, y: 0 },  
                placement: "far"  
            }]  
        }]  
    });
```

```
},
{
  type: "minorinterval", width: 1, height: 6,
  color: "#206BA4", distanceFromScale: { x: -27, y: 0 },
  placement: "far"
}]
}]
});
});
```

Execute the above code to render the following output.



### Adding multiple scales

You can set multiple scales for a single **Linear Gauge** control by using an array of scale objects. Each scale object is independent of each other. Refer the following code example to add multiple scale collection.

#### HTML

```
<div id="LinearGauge1"></div>
```

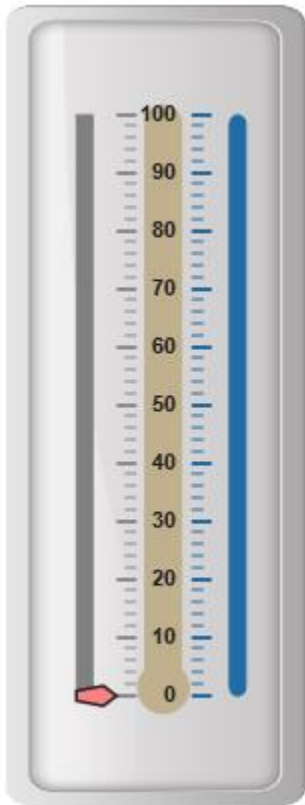
#### JAVASCRIPT

```
$(function () {
  // For Linear Gauge rendering
  var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
    enableAnimation: false,
    frame: { backgroundImageUrl: "../images/gauge/Gauge_linear_light.png" },
```



```
//Adding Scale collection
scales: [
//Adding Scale 1
{
width: 8,
position: { x: 15, y: 50 },
backgroundColor: "Grey",
border: { color: "Grey", width: 1 },
showMarkerPointers: true, showBarPointers: false,
//Adding label collection
labels: [{ distanceFromScale: { x: 50, y: 0 } }],
//Adding marker pointer collection
markerPointers: [{
type: "pentagon", placement: "near",
length: 10, width: 20, distanceFromScale: 20,
backgroundColor: "#FE8282"
}],
//Adding tick collection
ticks: [{
type: "majorinterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 30, y: 0 }
},
{
type: "minorinterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 30, y: 0 }
}]
},
//Adding Scale 2
{
width: 8, direction: ej.datavisualization.LinearGauge.Directions.Clockwise,
position: { x: 90, y: 50 }, type: "roundedrectangle",
backgroundColor: "#206BA4",
border: { color: "#206BA4", width: 1 },
showMarkerPointers: false, showBarPointers: false, showLabels: false,
ticks: [{
type: "majorinterval", width: 2,
color: "#206BA4", distanceFromScale: { x: -27, y: 0 }, placement: "far"
},
{
type: "minorinterval", width: 1, height: 6,
color: "#206BA4", distanceFromScale: { x: -27, y: 0 },
placement: "far"
}]
},
//Adding Scale 3
{
width: 18, length: 300,
position: { x: 54, y: 50 }, type: "thermometer",
backgroundColor: "#C0B08E",
border: { color: "#C0B08E", width: 1 },
showMarkerPointers: false, showBarPointers: false,
showLabels: false, showTicks: false,
}]
});
});
```

Execute the above code to render the following output.



### Marker Pointers

**Marker Pointer** value points out the actual value set in the **Linear Gauge**. You can set values for various pointer attributes such as `value`, `type`, `length`, `width`, `border` and `color` in pointer collection. You can also customize the pointers to improve the appearance of gauge.

#### Adding marker pointer collection

You can add **Marker Pointer** collection directly to the scale object. To add pointer collection in a gauge control refer the following code example.

#### HTML

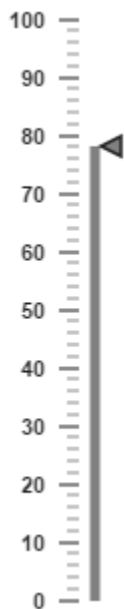
```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="../../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../../tsfiles/ej.web.all.d.ts"></reference>
module LinearGaugeComponent {
  $(function () {
    //For Linear gauge rendering
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      value: 78,
      //Adding scale collection
      scales: [{
        border: { color: "transparent", width: 0 },
        showBarPointers: true,
```

```
//Adding bar pointer collection
barPointers: [{ width: 5, backgroundColor: "Grey" }],
//Adding marker pointer collection
markerPointers: [{
  width: 10,
  length: 10,
  backgroundColor: "Grey",
  distanceFromScale: -12
}],
//Adding tick collection
ticks: [{
  type: "majorinterval", width: 2,
  color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
},
{
  type: "minorinterval", width: 1, height: 6,
  color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
}]
});
});
}
```

Execute the above code to render the following output.



#### Add marker pointer value

The **value** property is the important element in the marker pointer collection which indicates the gauge value. Real purpose of the **Linear Gauge** is based on the pointer value. You can set the pointer value either directly during rendering the control or it can be achieved by public method.

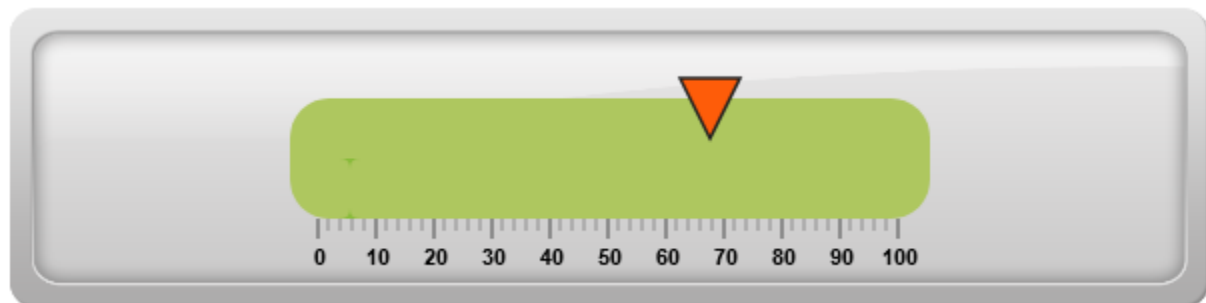
#### HTML

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
$(function () {  
  //For render linear gauge  
  var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
    enableAnimation: false,  
    width: 600,  
    height: 150,  
    orientation: "Horizontal",  
    labelColor: "Black",  
    enableResize: true,  
    //For Adding frame collection  
    frame: {  
      backgroundImageUrl: "../images/gauge/Gauge_linear_light1.png"  
    },  
    //For Adding scale collection  
    scales: [{  
      backgroundColor: "#AEC75F",  
      direction: ej.datavisualization.LinearGauge.Directions.Clockwise,  
      type: "roundedrectangle",  
      border: { color: "#AEC75F", width: 30 },  
      //Adding marker pointer collection  
      markerPointers: [{  
        width: 30, length: 30, backgroundColor: "#FE5C09",  
        distanceFromScale: 20, placement: "near",  
        value: 67.5  
      }],  
      //Adding label collection  
      labels: [{ angle: 90, distanceFromScale: { x: 0, y: 100 } }],  
      //Adding tick collection  
      ticks: [{  
        type: "majorinterval", width: 2,  
        color: "#8c8c8c", distanceFromScale: { x: 45, y: -1 }  
      },  
      {  
        type: "minorinterval", width: 1, height: 6,  
        color: "#8c8c8c", distanceFromScale: { x: 45, y: -1 }  
      }  
    ]  
  }];  
});
```

Execute the above code to render the following output.



## Pointer Styles

### Appearance

- Based on the value, the **pointer** points out the label value. You can set the pointer length and width using **length** and **width** property respectively. You can also adjust the opacity of the pointer using the **opacity** property which holds the value between 0 and 1. You can add the gradient effects to the pointer using **gradient** object.
- The marker pointer border is modified with the **border** object. It contains two border property namely **color** and **width** which are used to customize the border color of the scale and border width of the marker pointer. The background color can be customized with attribute **backgroundColor**.

### HTML

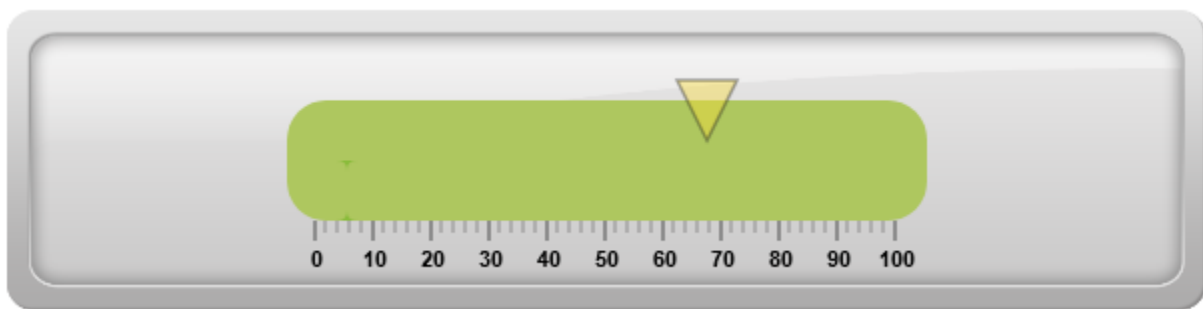
```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    // For Render Linear gauge  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        enableAnimation: false,  
        width: 600,  
        height: 150,  
        orientation: "Horizontal",  
        labelColor: "Black",  
        enableResize: true,  
        //Adding Frame object  
        frame: {  
            backgroundImageUrl: "../images/gauge/Gauge_linear_light1.png"  
        },  
        //Adding Scale collection  
        scales: [{  
            backgroundColor: "#AEC75F",  
            direction: ej.datavisualization.LinearGauge.Directions.Clockwise,  
            type: "roundedrectangle",  
            border: { color: "#AEC75F", width: 30 },  
            //Adding marker pointer collection  
            markerPointers: [{  
                //Adding width  
                width: 30,  
                //Adding height  
                length: 30,  
                //Adding opacity  
                opacity: 0.4,  
                //Adding background  
                backgroundColor: "#FCDD34",  
                distanceFromScale: 20,  
                placement: "near",  
                value: 67.5  
            }],  
            //Adding label collection  
            labels: [{ angle: 90, distanceFromScale: { x: 0, y: 100 } }],  
        }],  
    });
```

```
//Adding ticks collection
ticks: [{
  type: "majorinterval", width: 2,
  color: "#8c8c8c", distanceFromScale: { x: 45, y: -1 }
},
{
  type: "minorinterval", width: 1, height: 6,
  color: "#8c8c8c", distanceFromScale: { x: 45, y: -1 }
}]
});
```

Execute the above code to render the following output.



#### Positioning the pointer

- You can position the Pointer with two properties, **distanceFromScale** and **placement**. The **distanceFromScale** property defines the distance between the scale and pointer.
- The **placement** property is used to locate the pointer with respect to scale either inside or outside the scale or along the scale. It is an enumerable data type.

#### HTML

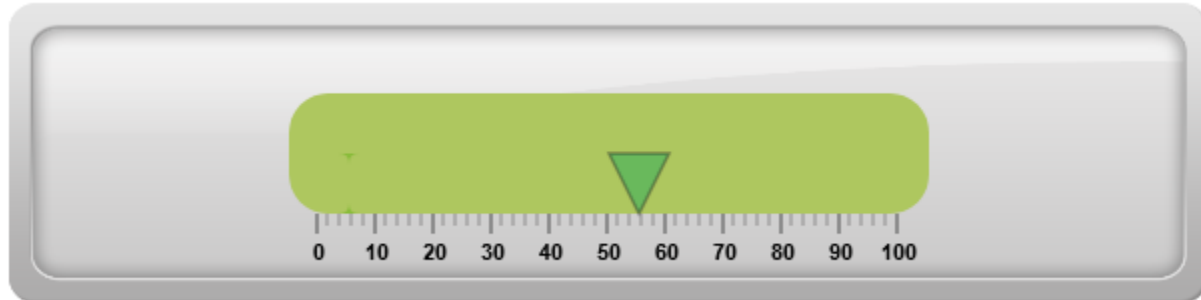
```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {
  //For Render Linear gauge
  var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
    enableAnimation: false,
    width: 600,
    height: 150,
    orientation: "Horizontal",
    labelColor: "Black",
    enableResize: true,
    //For Adding frame
    frame: {
      backgroundImageUrl: "../images/gauge/Gauge_linear_light1.png"
    },
    //For Adding Scale
    scales: [{
```

```
backgroundColor: "#AEC75F",
direction: ej.datavisualization.LinearGauge.Directions.Clockwise,
type: "roundedrectangle",
border: { color: "#AEC75F", width: 30 },
//For Adding marker pointer collection
markerPointers: [{
width: 30, height: 8, opacity: 0.4, backgroundColor: "#01A357",
distanceFromScale: 60,
placement: "near",
value: 55.5
}],
//For Adding label collection
labels: [{ angle: 90, distanceFromScale: { x: 0, y: 100 } }],
//For Adding tick collection
ticks: [{
type: "majorinterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 45, y: -1 }
},
{
type: "minorinterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 45, y: -1 }
}]
}];
});
```

Execute the above code to render the following output.



### Types

It is possible to change the dimension of the marker pointer. Dimensions available for marker pointer are,

- Rectangle
- Triangle
- Ellipse
- Diamond
- Pentagon
- Circle
- Slider
- Pointer
- Wedge
- Trapezoid

- Rounded Rectangle

### Multiple Marker Pointers

**Linear Gauge** can contain multiple pointers on it. You can use any combination and any number of pointers in a gauge. That is, a gauge can contain any number of marker pointer and any number of bar pointers. Refer the following code example containing multiple marker pointers.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {
  //For Render Linear gauge
  var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
    enableAnimation: false, // minimum: -10, maximum: 110,
    width: 600,
    height: 250,
    theme: "flatlight",
    orientation: "Horizontal",
    labelColor: "Black",
    enableResize: true,
    //For Adding frame
    frame: {
      backgroundImageUrl: "../images/gauge/Gauge_linear_light1.png"
    },
    //For Adding Scale collection
    scales: [{
      backgroundColor: "#AEC75F", showCustomLabels: true,
      direction: ej.datavisualization.LinearGauge.Directions.Clockwise,
      type: "roundedrectangle",
      border: { color: "#AEC75F", width: 30 },
      markerPointers: [
        // Adding marker pointer 1
        {
          width: 30, length: 30, backgroundColor: "#01A357",
          distanceFromScale: 60, placement: "near", value: 32.2
        },
        // Adding marker pointer 2
        {
          width: 10, length: 30, backgroundColor: "#90DAFB",
          distanceFromScale: 60, placement: "near", value: 23.7, type: "circle"
        },
        // Adding marker pointer 3
        {
          width: 3, length: 30, backgroundColor: "#90DAFB",
          distanceFromScale: 60, placement: "near", value: 23.7, type: "star"
        }
      ],
      // Adding label collection
      labels: [{ angle: 90, distanceFromScale: { x: 0, y: 100 } }],
      // Adding tick collection
      ticks: [{
        type: "majorinterval", width: 2,
        color: "#8c8c8c", distanceFromScale: { x: 45, y: -1 }
      }
    ]
  });
});
```

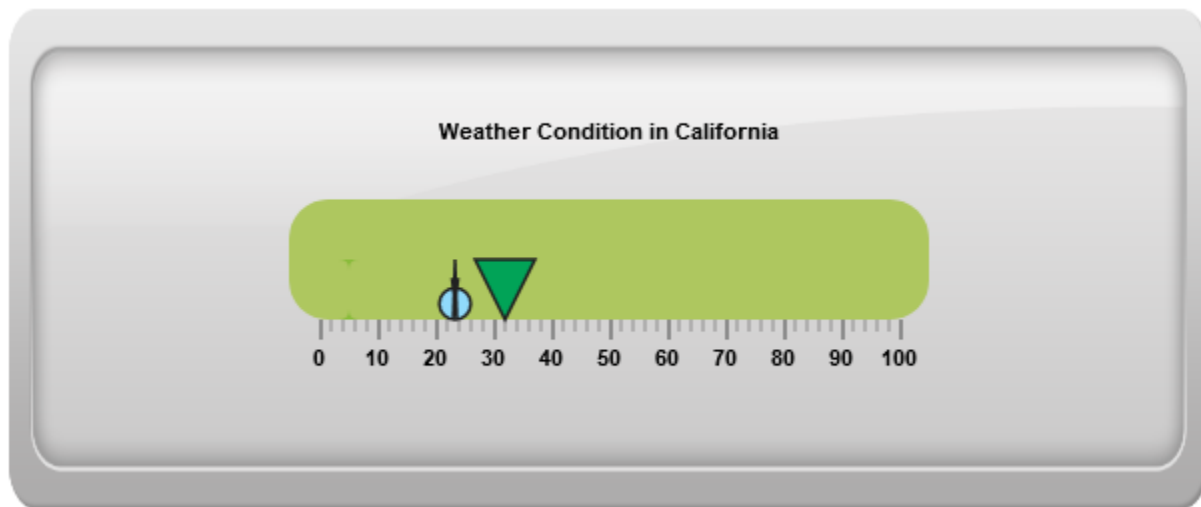


```

},
{
  type: "minorinterval", width: 1, height: 6,
  color: "#8c8c8c", distanceFromScale: { x: 45, y: -1 }
}],
// Adding custom label collection
customLabels: [{
  value: "Weather Condition in California", position: {
    x: 50, y: 20
  },
}]
});
});

```

Execute the above code to render the following output.



## Bar Pointers

**Bar Pointer** value points out the actual value set in the **Linear Gauge** as marker pointer. You can set the values of the various bar pointer attributes such as **value**, **width**, **border** and **color** in bar pointer collection. You can also customize the pointers to improve the appearance of gauge.

### Adding bar pointer collection

You can add Bar Pointer collection directly to the scale object. Refer the following code example.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

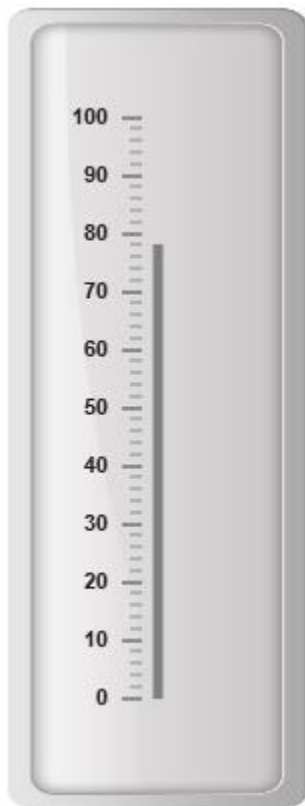
```

/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module LinearGaugeComponent {
  $ (function () {
    // For Rendering Linear gauge
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
    //For Adding frame

```

```
frame: {  
  innerWidth: 8,  
  outerWidth: 10,  
  backgroundImageUrl: "../images/gauge/Gauge_linear_light.png"  
},  
value: 78,  
//For Adding Scale collection  
scales: [{  
  backgroundColor: "transparent",  
  border: { color: "transparent", width: 0 },  
  showMarkerPointers: false, showBarPointers: true,  
  //For Adding bar pointer collection  
  barPointers: [{ width: 5, backgroundColor: "Grey"}],  
  //For Adding tick collection  
  ticks: [{ type: "majorinterval", width: 2,  
    color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 } },  
    { type: "minorinterval", width: 1,height:6,  
    color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 } }]  
  }]  
});  
});  
}
```

Execute the above code to render the following output.



### Adding bar pointer value

Bar pointer value is also important element in the **Linear Gauge** as it indicates the gauge value. Real purpose of the **Linear Gauge** is based on the pointer value. You can set the bar pointer **value** either directly during rendering the control or it can be achieved by public method.

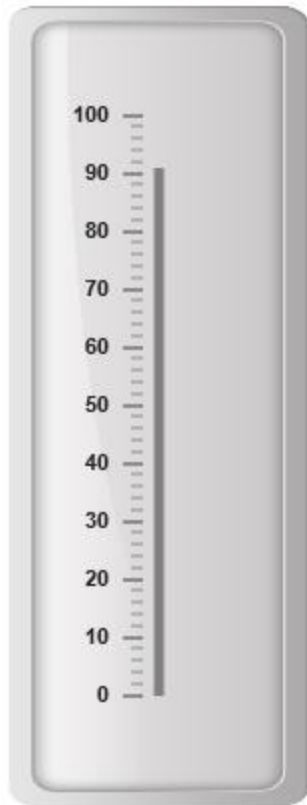
### HTML

```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    // For Rendering Linear gauge  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        enableAnimation: false,  
        //For Adding frame  
        frame: {  
            innerWidth: 8,  
            outerWidth: 10,  
            backgroundImageUrl: "../images/gauge/Gauge_linear_light.png"  
        },  
        //For Adding Scales  
        scales: [{  
            backgroundColor: "transparent",  
            border: { color: "transparent", width: 0 },  
            showMarkerPointers: false, showBarPointers: true,  
            //For Adding bar pointer collection  
            barPointers: [{  
                width: 5,  
                backgroundColor: "Grey",  
                value: 91  
            }],  
            //For Adding tick collection  
            ticks: [{  
                type: "majorinterval", width: 2,  
                color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }  
            },  
            {  
                type: "minorinterval", width: 1, height: 6,  
                color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }  
            }  
        ]  
        }  
    }  
});
```

Execute the above code to render the following output.



## Pointer Styles

### Appearance

- Based on the value, the bar pointer points out the label value. You can set the bar pointer width using `width` property and you can also adjust the opacity of the pointer using `opacity` property that holds the value between 0 and 1. You can add the gradient effects to the pointer using `gradient` object.
- The marker pointer border is modified with the object **border**. It has two border property, `color` and `width` which are used to customize the border color of the scale and border width of the marker pointer. The background color can be customized with attribute `backgroundColor`.

### HTML

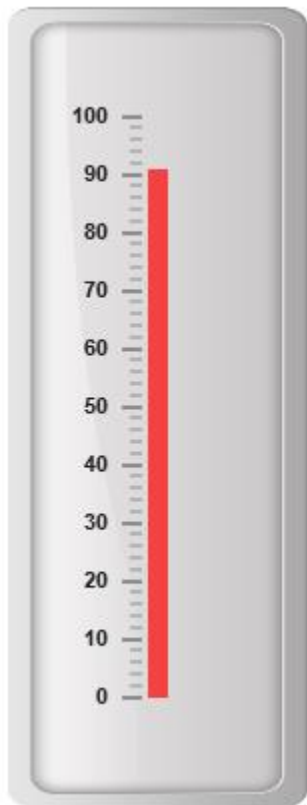
```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
  // For Rendering Linear gauge  
  var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
    enableAnimation: false,  
    // Adding Frame object  
    frame: {  
      innerWidth: 8,  
      outerWidth: 10,  
      backgroundImageUrl: "../images/gauge/Gauge_linear_light.png"  
    }  
  });  
});
```

```
},  
// Adding Scale collection  
scales: [{  
  backgroundColor: "transparent",  
  border: { color: "transparent", width: 0 },  
  showMarkerPointers: false, showBarPointers: true,  
  // Adding bar pointer collection  
  barPointers: [{  
    width: 10,  
    backgroundColor: "Red",  
    border: { color: "#860201", width: 2 },  
    opacity: 0.7,  
    value: 91 }],  
  // Adding tick collection  
  ticks: [{  
    type: "majorinterval", width: 2,  
    color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }  
  },  
  {  
    type: "minorinterval", width: 1, height: 6,  
    color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }  
  }]  
  }]  
});  
});
```

Execute the above code to render the following output.



### Positioning the pointer

- Bar pointer can be positioned with two properties such as **distanceFromScale** and **placement**. The **distanceFromScale** property defines the distance between the scale and pointer element.
- The **placement** property is used to locate the pointer with respect to scale either inside or outside the scale or along the scale. It is an enumerable data type.

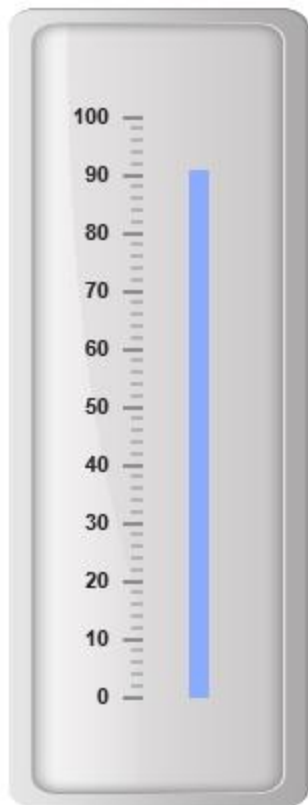
### HTML

```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
  // For Rendering Linear gauge  
  var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
    enableAnimation: false,  
    //Adding frame object  
    frame: {  
      innerWidth: 8,  
      outerWidth: 10,  
      backgroundImageUrl: "../images/gauge/Gauge_linear_light.png"  
    },  
    //Adding Scale collection  
    scales: [{  
      backgroundColor: "transparent",  
      border: { color: "transparent", width: 0 },  
      showMarkerPointers: false, showBarPointers: true,  
      //Adding bar pointer collection  
      barPointers: [{  
        width: 10,  
        backgroundColor: "#8BABFF",  
        value: 91,  
        placement: "near",  
        distanceFromScale: 20  
      }],  
      //Adding tick collection  
      ticks: [{  
        type: "majorinterval", width: 2,  
        color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }  
      },  
      {  
        type: "minorinterval", width: 1, height: 6,  
        color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }  
      }  
    ]  
  }]  
});
```

Execute the above code to render the following output.



### Multiple Bar Pointers

**Linear Gauge** can contain multiple bar pointers on it. You can use any combination and any number of pointers in a gauge. That is, a Gauge can contain any number of marker pointer and any number of bar pointers. Refer the following code example containing multiple bar pointers.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

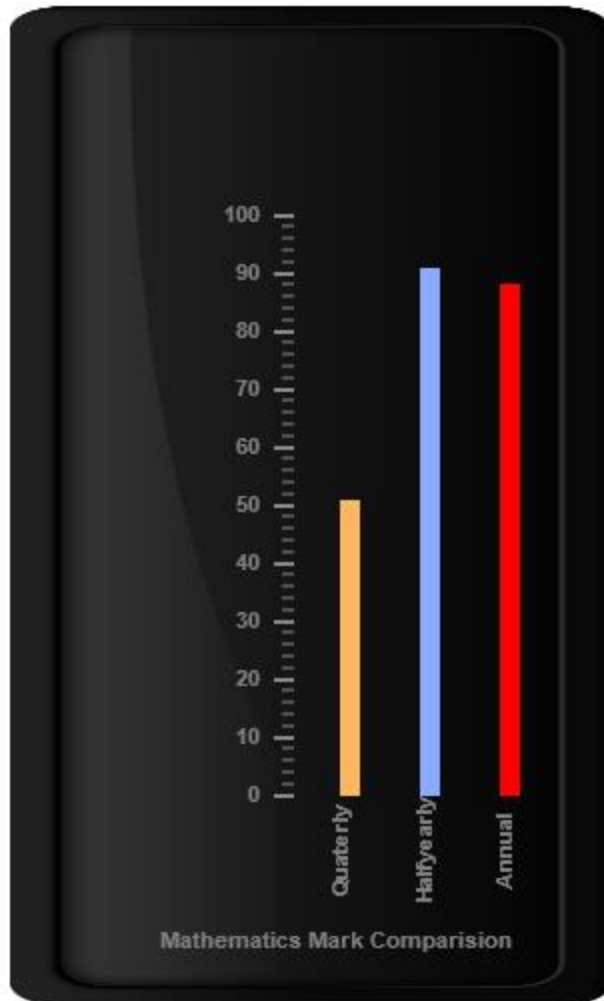
```
$(function () {  
    // For Rendering Linear gauge  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        enableAnimation: false, height: 500,  
        width: 300, labelColor: "Grey",  
        //Adding frame object  
        frame: {  
            innerWidth: 8,  
            outerWidth: 10,  
            backgroundImageUrl: "../images/gauge/Gauge_linear_dark1.png"  
        },  
        //Adding scale collection  
        scales: [{  
            backgroundColor: "transparent",  
            border: { color: "transparent", width: 0 },  
            showMarkerPointers: false, showBarPointers: true, showCustomLabels: true,  
            //Adding bar pointer collection  
            barPointers: [  

```

```
//Adding bar pointer 1
{
width: 10, backgroundColor: "#8BABFF",
value: 91, placement: "near", distanceFromScale: 60
},
//Adding bar pointer 2
{
width: 10, backgroundColor: "#FDB761", value: 51,
placement: "near", distanceFromScale: 20
},
//Adding bar pointer 3
{
width: 10, backgroundColor: "Red", value: 88,
placement: "near", distanceFromScale: 100
}
],
//Adding tick collection
ticks: [{
type: "majorinterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
},
{
type: "minorinterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
}],
//Adding custom label collection
customLabels: [
{
value: "Mathematics Mark Comparison",
position: { x: 55, y: 97 }
},
{ value: "Half yearly", position: { x: 72, y: 87 }, textAngle: 90 },
{ value: "Quarterly", position: { x: 56, y: 87 }, textAngle: 90 },
{ value: "Annual", position: { x: 87, y: 87 }, textAngle: 90 }
]
});
});
```

Execute the above code to render the following output.





## Labels

Labels are units that are used to display the values in the scales. You can customize Labels with the properties like `angle`, `color`, `font`, `opacity`, etc.

### Label Customization

#### Appearance

- The attribute **angle** is used to display the labels in the specified angles and **color** attribute is used to display the labels in specified color. You can adjust the opacity of the label with the property **opacity** and the values of it lies between 0 and 1. The `includeFirstValue` is a special property by enabling this property, the first value of the label is not rendered.
- Font option is also available on the labels. The basic three properties of fonts such as `size`, `family` and `style` can be achieved by **size**, **fontStyle** and **fontFamily**. Labels are two types such as major and minor. Major type labels are for major interval values and minor type labels are for minor interval values.

## HTML

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module LinearGaugeComponent {
$(function () {
// For Rendering Linear gauge
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
enableAnimation: false, value: 40,
//Adding Frame Object
frame: {
innerWidth: 8,
outerWidth: 10,
backgroundImageUrl: "../../images/gauge/Gauge_linear_light.png"
},
//Adding scale collection
scales: [{
backgroundColor: "transparent",
border: { color: "transparent", width: 0 },
showBarPointers: true, showCustomLabels: true, showMarkerPointers: false,
//Adding bar pointer collection
barPointers: [{ width: 10 }],
//Adding label collection
labels: [{
font: { size: "12px", fontFamily: "Arial", fontStyle: "Bold" },
textColor: "Red",
angle: 10,
opacity: 0.5,
includeFirstValue: false
}],
//Adding ticks collection
ticks: [{
type: "majorinterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
},
{
type: "minorinterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
}]
}
});
});
}
```

Execute the above code to render the following output.



### Unit text and Positioning

- The `unitText` property is used to add some text along with the labels. For example, in speedometer, you need to mention the units in kph. You can also add the unit text in front of the labels. To achieve this use the enumerable property `unitTextPlacement`.
- Labels can be positioned with the help of two properties such as **`distanceFromScale`** and **`placement`**. `distanceFromScale` property defines the distance between the scale and labels. `placement` property is used to locate the labels with respect to scale either inside the scale or outside the scale or along the scale. It is an enumerable data type.

### HTML

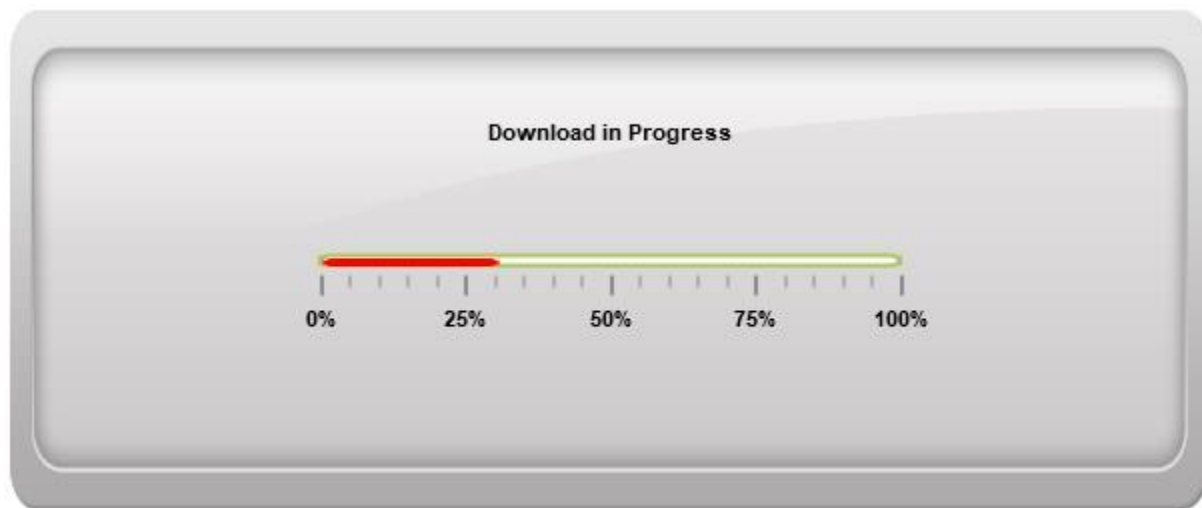
```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
$ (function () {  
  // For Linear Gauge rendering  
  var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
    enableAnimation: false, // minimum: -10, maximum: 110,  
    width: 600, value: 31,  
    height: 250,  
    theme: "flatlight",  
    orientation: "Horizontal",  
    labelColor: "Black",  
    enableResize: true,  
  });  
});
```

```
// Adding frame object
frame: {
  backgroundImageUrl: "../images/gauge/Gauge_linear_light1.png"
},
// Adding scale collection
scales: [{
  width: 5, majorIntervalValue: 25, minorIntervalValue: 5,
  backgroundColor: "White", showCustomLabels: true,
  showMarkerPointers: false, showBarPointers: true,
  direction: ej.datavisualization.LinearGauge.Directions.Clockwise,
  type: "roundedrectangle",
  border: { color: "#AEC75F", width: 2 },
  // Adding bar pointer collection
  barPointers: [{ width: 4, backgroundColor: "Red" }],
  // Adding label collection
  labels: [{
    angle: 90,
    distanceFromScale: { x: 0, y: 60 },
    unitText: "%"
  }],
  // Adding tick collection
  ticks: [{
    type: "majorinterval", width: 2,
    color: "#8c8c8c", distanceFromScale: { x: 0, y: 25 }
  },
  {
    type: "minorinterval", width: 1, height: 6,
    color: "#8c8c8c", distanceFromScale: { x: 0, y: 25 }
  }],
  customLabels: [{
    value: "Download in Progress", position: { x: 50, y: 20 },
  }],
}];
});
```

Execute the above code to render the following output.



## Ticks

Ticks are used to mark some values on the scale. Based on the tick's value you can set the labels on the required position.

### Adding tick collection

Tick collection can be directly added to the scale object. Refer the following code example to add tick collection in a **Linear Gauge** control.

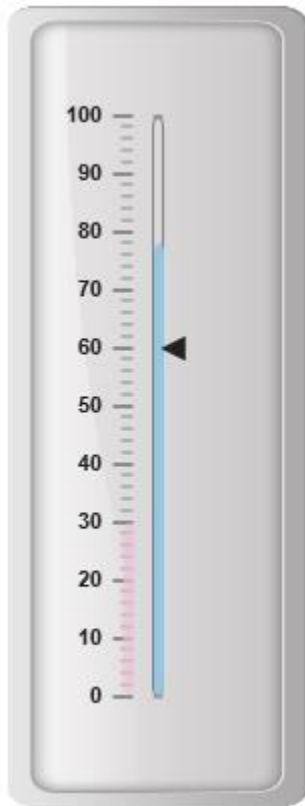
#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module LinearGaugeComponent {
$(function () {
//For Linear gauge rendering
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
enableAnimation: false,
//Adding frame object
frame: {
innerWidth: 8,
outerWidth: 10,
backgroundImageUrl: "../../images/gauge/Gauge_linear_light.png"
},
value: 78,
//Adding scale collection
scales: [{
width: 5,
backgroundColor: "transparent", type: "roundedrectangle",
border: { color: "Grey", width: 1 }, showBarPointers: true,
//Adding label collection
labels: [{ distanceFromScale: { x: -25, y: 0 } }],
//Adding marker pointer collection
markerPointers: [{ width: 10, length: 10, value: 60 }],
//Adding bar pointer collection
barPointers: [{ width: 5, backgroundColor: "#95C7E0" },
{
width: 6, backgroundColor: "#EDC1D7",
distanceFromScale: -15, value: 30, opacity: 0.7
}],
//Adding tick collection
ticks: [{
type: "majorinterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: -10, y: 0 }, position: "far"
},
{
type: "minorinterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: -10, y: 0 }, position: "far"
}]
}];
});
}
```

Execute the above code to render the following output.



### Tick Customization

#### Appearance

- Height and width of the ticks can be applied by using the properties `height` and `width`. You can customize ticks with the properties like `angle`, `color`, etc. `angle` attribute is used to display the labels in the specified angles and `color` attribute is used to display the labels in specified color.
- Ticks are two types such as major and minor. The opacity of the labels can be adjusted with the property `opacity`. The opacity values lies between 0 and 1.

#### HTML

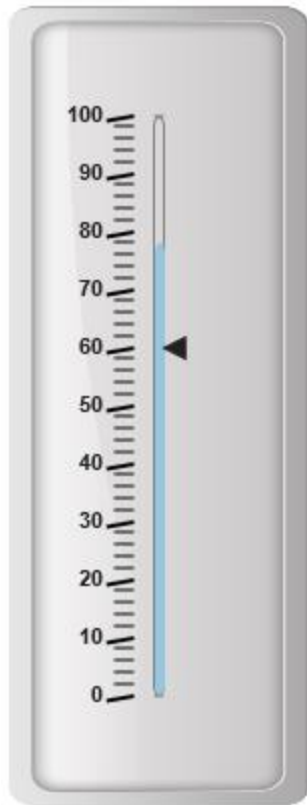
```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {  
  //For Linear gauge rendering  
  var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
    //Adding scale object  
    frame: {  
      innerWidth: 8,  
      outerWidth: 10,  
      backgroundImageUrl: "../images/gauge/Gauge_linear_light.png"  
    }  
  });  
});
```

```
},
value: 78,
//Adding scale collection
scales: [{
width:5,
backgroundColor: "transparent", type: "roundedrectangle",
border: { color: "Grey", width: 1 },showBarPointers: true,
//Adding label collection
labels: [{ distanceFromScale: { x: -25, y: 0 } }],
//Adding marker pointer collection
markerPointers:[{width:10,length:10, value:60}],
//Adding bar pointer collection
barPointers: [{ width: 5, backgroundColor: "#95C7E0" }],
//Adding ticks collection
ticks: [{
type: "majorinterval",
width: 2,
height:14,
angle:10,
color: "Black",
distanceFromScale: { x: -10, y: 0 },position:"far"
},
{
type: "minorinterval",
width: 1,
height: 10,
opacity:0.5,
color: "Black",
distanceFromScale: { x: -10, y: 0 }, position: "far"
}]
}];
});
```

Execute the above code to render the following output.



## Types

Ticks are two types such as **majorInterval** and **minorInterval**. Major type ticks are for major interval values and minor type ticks are for minor interval values.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
$(function () {
    //For Linear gauge rendering
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
        enableAnimation: false,
        //Adding frame object
        frame: {
            innerWidth: 8,
            outerWidth: 10,
            backgroundImageUrl: "../images/gauge/Gauge_linear_light.png"
        },
        value: 78,
        //Adding scale collection
        scales: [{
            width: 5,
            backgroundColor: "transparent", type: "roundedrectangle",
            border: { color: "Grey", width: 1 }, showBarPointers: true,
            //Adding label collection
            labels: [{ distanceFromScale: { x: -25, y: 0 } }],
            //Adding marker pointer collection
```



```
markerPointers: [{ width: 10, length: 10, value: 60 }],  
//Adding bar pointer collection  
barPointers: [{ width: 5, backgroundColor: "#95C7E0" }],  
//Adding tick collection  
ticks: [{  
  type: "majorinterval", width: 2, height: 14,  
  color: "Black", position: "far"  
},  
{  
  type: "minorinterval",  
}]  
}];  
});
```

Execute the above code to render the following output.



### Positioning the ticks

- You can position ticks with the help of two properties such as **distanceFromScale** and **placement**. The property **distanceFromScale** defines the distance between the scale and ticks.
- **placement** property is used to locate the ticks with respect to scale either inside the scale or outside the scale or along the scale. It is an enumerable data type.

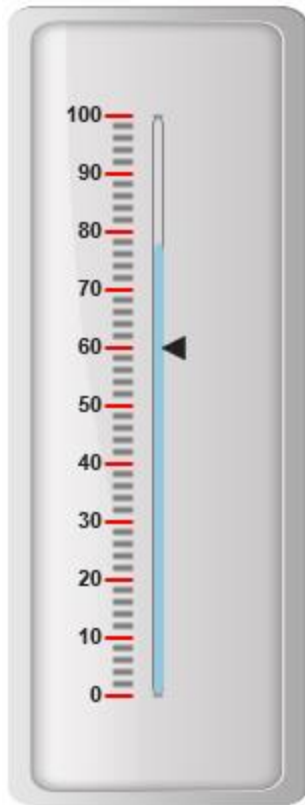
### HTML

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
$(function () {  
  // For Linear Gauge rendering  
  var sample = new  
  ej.datavisualization.LinearGauge($("#LinearGauge1"), {enableAnimation:false,  
  //Adding frame collection  
  frame: {  
    innerWidth: 8,  
    outerWidth: 10,  
    backgroundImageUrl: "../images/gauge/Gauge_linear_light.png"  
  },  
  value: 78,  
  //Adding scale collection  
  scales: [{width:5,  
    backgroundColor: "transparent", type: "roundedrectangle",  
    border: { color: "Grey", width: 1 },showBarPointers: true,  
    //Adding label collection  
    labels: [{ distanceFromScale: { x: -25, y: 0 } }],  
    //Adding marker pointer collection  
    markerPointers:[{width:10,length:10, value:60}],  
    //Adding bar pointer collection  
    barPointers: [{ width: 5, backgroundColor: "#95C7E0" }],  
    //Adding tick collection  
    ticks: [{  
      type: "majorinterval", width: 2,height:14,  
      color: "Black", distanceFromScale: { x: -10, y: 0 },  
      position: "far", color: "Red"  
    },  
    {  
      type: "minorinterval",distanceFromScale: { x: -10, y: 0 },  
      color:"grey"  
    }  
  ]  
  }  
  }  
});
```

Execute the above code to render the following output.



## Ranges

Ranges are used to specify or group the scale values. You can describe the values in the pointers using ranges.

### Adding range collection

Range collection can be directly added to the scale object. Refer the following code example to add range collection in a **Linear Gauge** control.

### HTML

```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

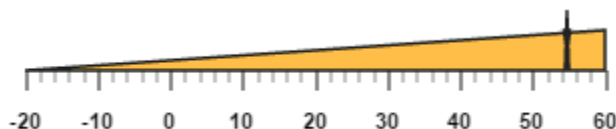
```
/// <reference path="../../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../../tsfiles/ej.web.all.d.ts"></reference>
module LinearGaugeComponent {
    $(function () {
        //For Linear gauge rendering
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            enableAnimation: false,
            width: 600,
            height: 150,
            orientation: "Horizontal",
            labelColor: "Black",
            enableResize: true,
            //Adding scale collection
            scales: [{
```

```

width: 0,
backgroundColor: "#AEC75F",
direction: ej.datavisualization.LinearGauge.Directions.Clockwise,
border: { width: 0, color: "transparent" }, minimum: -20, maximum: 60,
showBarPointers: false, showRanges: true,
//Adding marker pointers collection
markerPointers: [{
width: 3, length: 30, backgroundColor: "#FE5C09", type: "star",
distanceFromScale: 20, placement: "near",
value: 55
}],
//Adding label collection
labels: [{ angle: 90, distanceFromScale: { x: 0, y: 50 } }],
//Adding tick collection
ticks: [{
type: "majorinterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 25, y: -1 }
},
{
type: "minorinterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 25, y: -1 }
}],
//Adding range collection
ranges: [{
startValue: -20, endValue: 60,
startWidth: 0, endWidth: 20, backgroundColor: "#FEBE48",
placement: "near", distanceFromScale: 20
}]
}];
});
}

```

Execute the above code to render the following output.



## Range Customization

### Appearance

The major attributes for ranges are **startValue** and **endValue**. The **startValue** defines the start position of the range and **endValue** defines the end position of the range. The **startWidth** and **endWidth** are used to specify the range width at the starting and ending position of the ranges.

### HTML

```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

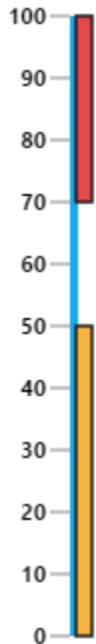
```

$(function () {
// For Linear Gauge rendering
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {

```

```
labelColor: "#8c8c8c", width: 500, load: "loadGaugeTheme",
//Adding scale collection
scales: [{
position: { x: 50, y: 50 },
width: 4, backgroundColor: "#10ADF5", border: {
color:
"transparent", width: 0
}, showRanges: true, showScaleBar: true,
showMarkerPointers: false, length: 310,
//Adding label collection
labels: [{
font: {
size: "11px", fontFamily: "Segoe UI", fontStyle:
"bold"
}, distanceFromScale: { x: -12 }
}],
//Adding ticks collection
ticks: [{ type: "majorinterval", width: 1, color: "#8c8c8c" }],
//Adding ranges collection
ranges: [
{
endValue: 50, // For setting range end value
startValue: 0, //For setting range start value
startWidth: 8,
endWidth: 8,
backgroundColor: "#F6B53F",
distanceFromScale: 5
},
{
endValue: 100,
startValue: 70,
startWidth: 8,
endWidth: 8,
distanceFromScale: 5,
backgroundColor: "#E94649" //For setting range background color
}]
}];
});
```

Execute the above code to render the following output.



### Colors and Border

- You can customize the ranges to improve the appearance of the **Gauge**. The range border is modified with the object called **border**. It has two border property such as **color** and **width** which are used to customize the border color of the ranges and border width of the ranges.
- You can set the background color to improve the look and feel of the **Linear Gauge**. For customizing the background color of the ranges, **backgroundColor** is used. You can add the gradient effects to the ranges by using **gradient** object.

### HTML

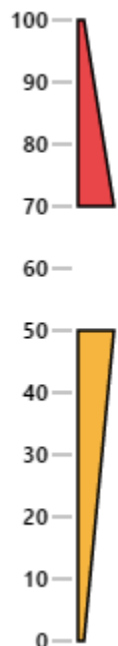
```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    // For Linear gauge rendering  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        labelColor: "#8c8c8c", width: 500, load: "loadGaugeTheme",  
        //Adding scale collection  
        scales: [{  
            position: { x: 50, y: 50 },  
            //Adding label collection  
            labels: [{  
                font: {  
                    size: "11px", fontFamily: "Segoe UI", fontStyle:  
                        "bold"  
                }, distanceFromScale: { x: -12 }  
            }],  
            //Adding ticks collection  
            ticks: [{ type: "majorinterval", width: 1, color: "#8c8c8c" }],  
            width: 4, backgroundColor: "transparent", border: {
```

```
color:
"transparent", width: 0
}, showRanges: true, showScaleBar: true,
showMarkerPointers: false, length: 310,
//Adding ranges collection
ranges: [{
  endValue: 50, // For setting range end value
  startValue: 0, // For setting range start value
  backgroundColor: "#F6B53F",
  border: { color: "black" },
  startWidth: 3,
  endWidth: 18,
  distanceFromScale: 10
}, {
  endValue: 100,
  startValue: 70,
  backgroundColor: "#E94649",
  border: { color: "black" },
  startWidth: 18,
  endWidth: 3,
  distanceFromScale: 10
}]
});
});
```

Execute the above code to render the following output.



### Positioning the ranges

- You can position ranges using two properties such as **distanceFromScale** and **placement**. The **distanceFromScale** property defines the distance between the scale and range.

- **placement** property is used to locate the pointer with respect to scale either inside the scale or outside the scale or along the scale. It is an enumerable data type.

## HTML

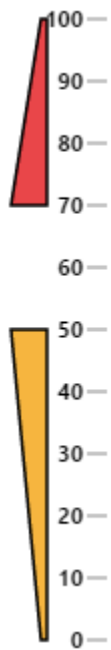
```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
$(function () {  
    // For Linear Gauge rendering  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        labelColor: "#8c8c8c", width: 500, load: "loadGaugeTheme",  
        //Adding scale position  
        scales: [{  
            position: { x: 50, y: 50 },  
            //Adding label collection  
            labels: [{  
                font: {  
                    size: "11px", fontFamily: "Segoe UI", fontStyle:  
                        "bold"  
                }, distanceFromScale: { x: -12 }  
            }],  
            //Adding ticks collection  
            ticks: [{ type: "majorinterval", width: 1, color: "#8c8c8c" }],  
            width: 4, backgroundColor: "transparent", border: {  
                color:  
                    "transparent", width: 0  
            }, showRanges: true, showScaleBar: true,  
            showMarkerPointers: false, length: 310,  
            //Adding ranges collection  
            ranges: [{  
                endValue: 50, // For setting range end value  
                startValue: 0, // For setting range start value  
                backgroundColor: "#F6B53F",  
                border: { color: "black" },  
                startWidth: 3,  
                endWidth: 18,  
                distanceFromScale: -30,  
                placement: "near"  
            }  
            , {  
                endValue: 100,  
                startValue: 70,  
                backgroundColor: "#E94649",  
                border: { color: "black" },  
                startWidth: 18,  
                endWidth: 3,  
                distanceFromScale: -30,  
                placement: "near"  
            }  
        ]  
    }]  
});  
});
```



Execute the above code to render the following output.



### Multiple Ranges

You can set multiple ranges by adding an array of range objects. Refer the following code example for multiple range functionality.

#### HTML

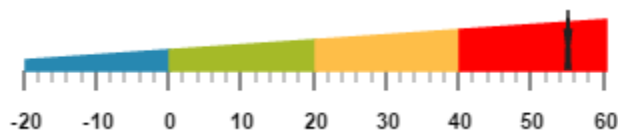
```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
$(function () {
    // For rendering Linear gauge
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
        enableAnimation: false,
        width: 600,
        height: 150,
        orientation: "Horizontal",
        labelColor: "Black",
        enableResize: true,
        //Adding scale collection
        scales: [{
            width: 0,
            backgroundColor: "#AEC75F",
            direction: ej.datavisualization.LinearGauge.Directions.Clockwise,
            border: { width: 0, color: "transparent" }, minimum: -20, maximum: 60,
            showBarPointers: false, showRanges: true,
            //Adding marker pointer collection
            markerPointers: [{
                width: 3, length: 30, backgroundColor: "#FE5C09", type: "star",
                distanceFromScale: 20, placement: "near",
                value: 55
            }],
        }],
    });
});
```

```
//Adding label collection
labels: [{ angle: 90, distanceFromScale: { x: 0, y: 50 } }],
//Adding tick collection
ticks: [{
  type: "majorinterval", width: 2,
  color: "#8c8c8c", distanceFromScale: { x: 20, y: -1 }
},
{
  type: "minorinterval", width: 1, height: 6,
  color: "#8c8c8c", distanceFromScale: { x: 20, y: -1 }
}],
//Adding ranges collection
ranges: [
  //Adding range 1
  {
    placement: "near",
    distanceFromScale: 20,
    startValue: -20, endValue: 0, startWidth: 5, endWidth: 10,
    backgroundColor: "#2788B1", border: { color: "#2788B1" }
  },
  //Adding range 2
  {
    placement: "near",
    distanceFromScale: 20,
    startValue: 0, endValue: 20, startWidth: 10, endWidth: 15,
    backgroundColor: "#A5BA28", border: { color: "#A5BA28" }
  },
  //Adding range 3
  {
    placement: "near",
    distanceFromScale: 20,
    startValue: 20, endValue: 40, startWidth: 15, endWidth: 20,
    backgroundColor: "#FEBE48", border: { color: "#FEBE48" }
  },
  //Adding range 4
  {
    placement: "near",
    distanceFromScale: 20,
    startValue: 40, endValue: 60, startWidth: 20, endWidth: 25,
    backgroundColor: "Red", border: { color: "Red" }
  }
]
});
});
```

Execute the above code to render the following output.



### Custom labels

Custom labels are the text that can paste in any location of the **Linear Gauge**. It is used to define the purpose of the gauge.

### Adding Custom label collection

Custom labels collection can be directly added to the scale object. Refer the following code to add custom labels collection in a **Linear Gauge** control.

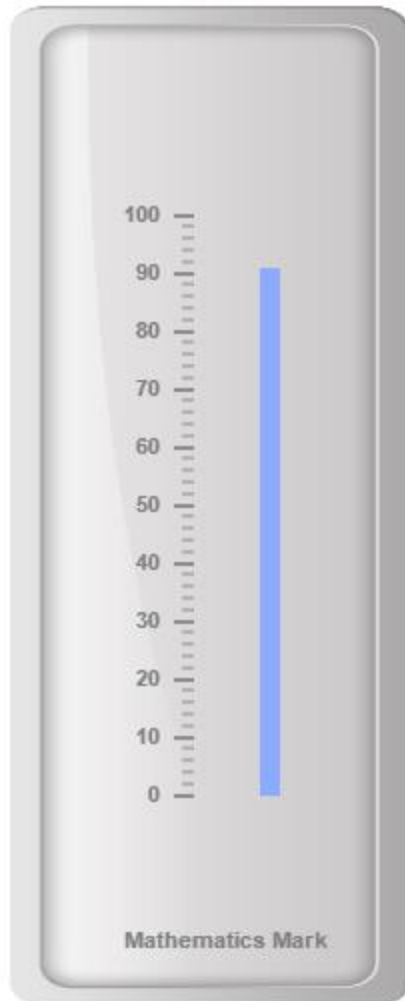
#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module LinearGaugeComponent {
$(function () {
// For Linear Gauge rendering
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
enableAnimation: false, height: 500, width: 200, labelColor: "Grey",
//Adding frame object
frame: {
innerWidth: 8,
outerWidth: 10,
backgroundImageUrl: "../../images/gauge/Gauge_linear_light.png"
},
//Adding scale collection
scales: [{
backgroundColor: "transparent",
border: { color: "transparent", width: 0 },
showMarkerPointers: false, showBarPointers: true,
showCustomLabels: true,
//Adding bar pointer collection
barPointers: [{
width: 10, backgroundColor: "#8BABFF",
value: 91, placement: "near", distanceFromScale: 30
}],
//Adding tick collection
ticks: [{
type: "majorinterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
},
{
type: "minorinterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
}],
//Adding custom label collection
customLabels: [{
value: "Mathematics Mark", position: { x: 55, y: 97 }
}]
}];
});
}
```

Execute the above code to render the following output.



## Basic Customization

### Appearance

- You can customize custom labels using the properties like `textAngle`, `color` and `font`. The API `textAngle` is used to display the custom labels in the specified angles and `color` attribute is used to display the custom labels in specified color. You can use `value` attribute to set the text value in the custom labels.
- To display the custom labels, set `showCustomLabels` as 'true'. Font option is also available on the custom labels. The basic three properties of fonts such as `size`, `family` and `style` can be achieved by `size`, `fontStyle` and `fontFamily`. You can adjust the opacity of the label with the property `opacity` and the value of opacity lies between 0 and 1.

### HTML

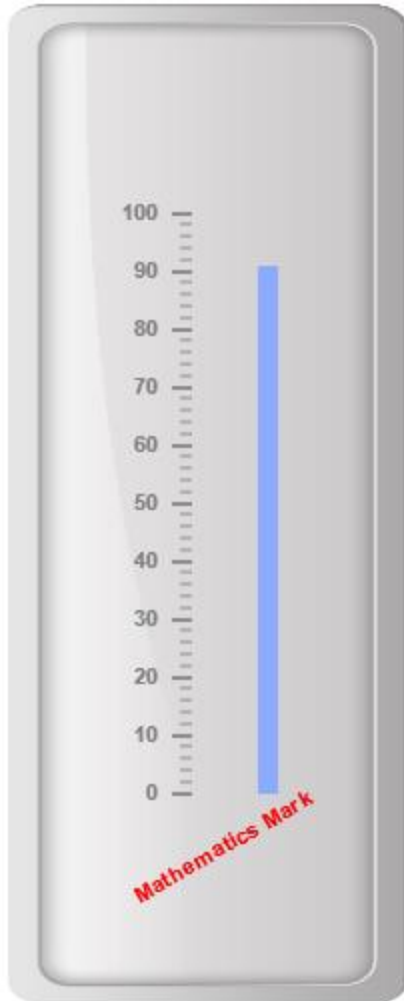
```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
$(function () {
```

```
// For Linear Gauge rendering
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
  enableAnimation: false, height: 500, width: 200, labelColor: "Grey",
  //Adding frame collection
  frame: {
    innerWidth: 8,
    outerWidth: 10,
    backgroundImageUrl: "../images/gauge/Gauge_linear_light.png"
  },
  //Adding scale collection
  scales: [{
    backgroundColor: "transparent",
    border: { color: "transparent", width: 0 },
    showMarkerPointers: false, showBarPointers: true,
    showCustomLabels: true,
    //Adding bar pointer collection
    barPointers: [
      {
        width: 10, backgroundColor: "#8BABFF",
        value: 91, placement: "near", distanceFromScale: 30
      }
    ],
    //Adding tick collection
    ticks: [{
      type: "majorinterval", width: 2,
      color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
    },
    {
      type: "minorinterval", width: 1, height: 6,
      color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
    }
  ]],
  //Adding custom labels
  customLabels: [{
    position: { x: 55, y: 87 },
    value: "Mathematics Mark",
    color: "Red",
    textAngle: 30,
    opacity: 0.5
  }]
}]);
```

Execute the above code to render the following output.



### Locating the CustomLabels

To set the location of the custom label in **Linear Gauge**, **position** property is used. You can position the custom labels in horizontal and vertical axis using **x** and **y** axis respectively.

### HTML

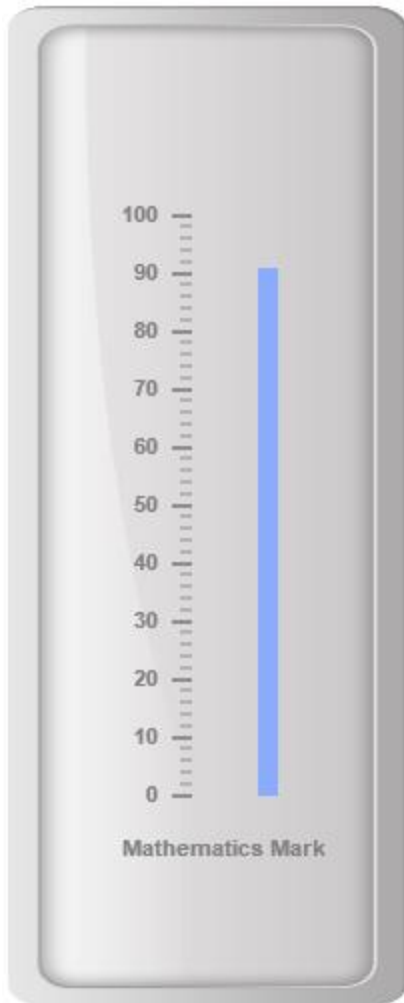
```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    //For rendering Liner gauge  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        enableAnimation: false, height: 500, width: 200, labelColor: "Grey",  
        //Adding frame object  
        frame: {  
            innerWidth: 8,  
            outerWidth: 10,  
            backgroundImageUrl: "../images/gauge/Gauge_linear_light.png"  
        },  
        //Adding scale collection  
        scales: [{
```

```
backgroundColor: "transparent",
border: { color: "transparent", width: 0 },
showMarkerPointers: false, showBarPointers: true,
showCustomLabels: true,
//Adding bar pointer collection
barPointers: [
{
width: 10, backgroundColor: "#8BABFF",
value: 91, placement: "near", distanceFromScale: 30
}
],
//Adding ticks collection
ticks: [{
type: "majorinterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
},
{
type: "minorinterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
}],
//Adding custom label collection
customLabels: [
{ value: "Mathematics Mark", position: { x: 55, y: 87 } }]
];
});
```

Execute the above code to render the following output.



### Multiple Custom Labels

You can set multiple custom labels in a single **Linear Gauge** by adding an array of custom label objects. Refer the following code example for multiple custom label functionality.

#### HTML

```
<div id="LinearGauge1"></div>
```

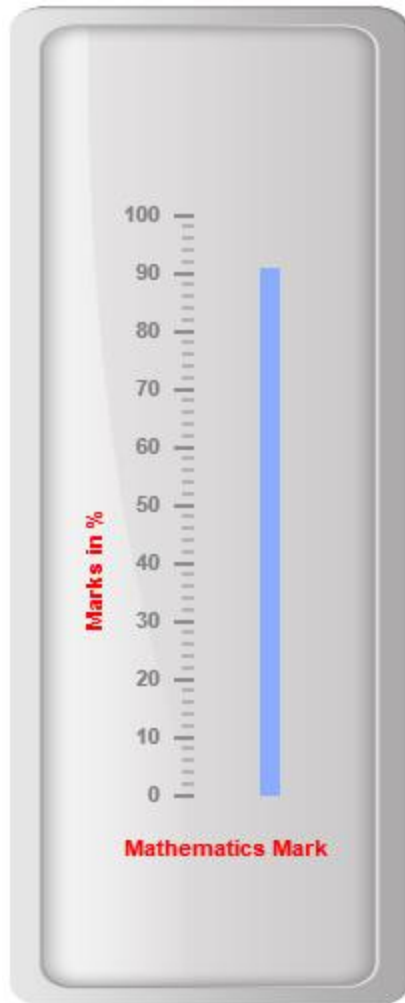
#### JAVASCRIPT

```
$(function () {  
    // For Linear Gauge rendering  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        enableAnimation: false, height: 500, width: 200, labelColor: "Grey",  
        //Adding frame object  
        frame: {  
            innerWidth: 8,  
            outerWidth: 10,  
            backgroundImageUrl: "../images/gauge/Gauge_linear_light.png"  
        },  
        //Adding scale collection  
        scales: [{
```



```
backgroundColor: "transparent",
border: { color: "transparent", width: 0 },
showMarkerPointers: false, showBarPointers: true,
showCustomLabels: true,
//Adding bar pointer collection
barPointers: [
{
width: 10, backgroundColor: "#8BABFF",
value: 91, placement: "near", distanceFromScale: 30
}
],
//Adding ticks collection
ticks: [{
type: "majorinterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
},
{
type: "minorinterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
}],
//Adding custom label collection
customLabels: [
{
value: "Mathematics Mark", position: { x: 55, y: 87 },
color: "Red"
},
{
value: "Marks in %", position: { x: 15, y: 57 },
color: "Red", textAngle: 90
}
]
});
});
```

Execute the above code to render the following output.



## Indicators

Indicators simply indicates the current status of the pointer. Indicators are in several formats such as in shape format, textual format and image format.

### Setting Dimension

- You can enable indicators by setting `showIndicators` to 'true' in scale collection. The `height` and `width` property for the indicators are used to specify the area allocated to the indicator for the width and height respectively.
- You can use the position collection to `position` the indicators along **X** and **Y** axis. `x` specifies horizontal position in indicators whereas `y` specifies vertical position in indicators. Indicators are of several types such as, dimensions like circle, rectangle, rounded rectangle, text and image. By using the `type` property it can be applied. For image type `imageUrl` property is used.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module LinearGaugeComponent {
$(function () {
// For Linear Gauge rendering
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
enableAnimation: false,
//Adding frame object
frame: { backgroundImageUrl: "../images/gauge/Gauge_linear_light.png" },
value: 78,
//Adding scale collection
scales: [{
width: 0,
border: { color: "transparent", width: 0 },
minimum: 0,
maximum: 300,
minorIntervalValue: 5,
majorIntervalValue: 30,
showBarPointers: false,
showIndicators: true,
//Adding marker pointer collection
markerPointers: [{
width: 10, length: 10,
backgroundColor: "Grey", distanceFromScale: 12
}],
//Adding ticks collection
ticks: [{
type: "majorinterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
},
{
type: "minorinterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
}],
//Adding indicator collection
indicators: [{
height: 10,
width: 10,
type: "circle",
position: { x: 50, y: 100 }
}]
}];
});
}
```

Execute the above code to render the following output.



### State Ranges

State ranges are used to specify the indicator behavior in the certain region. `startValue` and `endValue` are used to set the range bound for the pointer. Whenever the pointer crosses the specified region, the indicator attributes are applied for the ranges.

### HTML

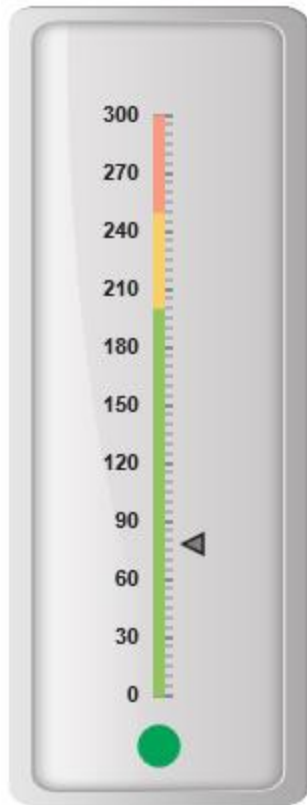
```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        enableAnimation: false, readOnly: false,  
        //Adding frame object  
        frame: { backgroundImageUrl: "../images/gauge/Gauge_linear_light.png" },  
        value: 78,  
        //Adding scale object  
        scales: [{  
            width: 0,  
            border: { color: "transparent", width: 0 },  
            minimum: 0,  
            maximum: 300,  
            minorIntervalValue: 5,  
            majorIntervalValue: 30,  
            showRanges: true,  
            showBarPointers: false,  
            showIndicators: true,  
        }],  
    });  
});
```

```
//Adding marker pointer collection
markerPointers: [{
width: 10, length: 10, backgroundColor: "Grey",
distanceFromScale: 12
}],
//Adding tick collection
ticks: [{
type: "majorinterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
},
{
type: "minorinterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
}],
//Adding range collection
ranges: [
{
startWidth: 5, endWidth: 5, startValue: 0, endValue: 200,
backgroundColor: "#94C361", border: {
color: "#94C361", width: 1
}
},
{
startWidth: 5, endWidth: 5, startValue: 200, endValue: 250,
backgroundColor: "#F9CF67", border: {
color: "#F9CF67", width: 1
}
},
{
startWidth: 5, endWidth: 5, startValue: 250, endValue: 300,
backgroundColor: "#F89B83", border: {
color: "#F89B83", width: 1
}
}
],
//Adding indicator collection
indicators: [
{
height: 10, width: 10, type: "circle", position: { x: 50, y: 100 },
//Adding State ranges collection
stateRanges: [
{
backgroundColor: "#02A258", endValue: 200,
startValue: 0, borderColor: "#02A258"
},
{
backgroundColor: "Grey", endValue: 300,
startValue: 200, borderColor: "Grey"
}
]
}
]
});
});
```

Execute the above code to render the following output.



Linear Gauge with indicator state ranges

### Color and Appearance

The `backgroundColor` and `borderColor` sets the appearance behavior for the indicators. You can apply this only if it lies within the state ranges. Otherwise default behavior will be applied.

The indicator border is modified with the border object. It contains two `border` property namely `color` and `width` which are used to customize the border color of the indicator and border width of the indicator pointer.

### HTML

```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    // For Linear Gauge rendering  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        enableAnimation: false,  
        //Adding frame object  
        frame: { backgroundImageUrl: "../images/gauge/Gauge_linear_light.png" },  
        value: 78,  
        //Adding scale object  
        scales: [{  
            width: 0,  
            border: { color: "transparent", width: 0 },  
            minimum: 0,  
            maximum: 300,
```

```
minorIntervalValue: 5,  
majorIntervalValue: 30,  
showBarPointers: false,  
showIndicators: true,  
//Adding marker pointer collection  
markerPointers: [{  
width: 10, length: 10,  
backgroundColor: "Grey", distanceFromScale: 12  
}],  
//Adding ticks collection  
ticks: [{  
type: "majorinterval", width: 2,  
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }  
},  
{  
type: "minorinterval", width: 1, height: 6,  
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }  
}],  
//Adding indicator collection  
indicators: [{  
height: 10, width: 10, type: "circle", position: { x: 50, y: 100 },  
stateRanges: [{  
backgroundColor: "#91B64E", endValue: 300,  
startValue: 0, borderColor: "#91B64E"  
}]  
}]  
});
```

Execute the above code to render the following output.



### Font options

The basic font options available for the textual type indicators in the **Linear Gauge** such as Size, font style and font family are achieved by the properties `size`, `fontStyle` and `fontFamily`.

To specifies the textLocation in bar indicators, you can use the `textLocation` property.

### HTML

```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
$(function () {  
    // For Linear Gauge rendering  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        enableAnimation: false, readOnly: false,  
        //Adding frame object  
        frame: { backgroundImageUrl: "../images/gauge/Gauge_linear_light.png" },  
        value: 78,  
        //Adding scale collection  
        scales: [{  
            width: 0,  
            border: { color: "transparent", width: 0 },  
            minimum: 0,  
            maximum: 300,  
            minorIntervalValue: 5,  
            majorIntervalValue: 30,  
            showRanges: true,  
            showBarPointers: false,  
        }],  
    });  
});
```



```
showIndicators: true,
//Adding marker pointer collection
markerPointers: [{
width: 10, length: 10,
backgroundColor: "Grey", distanceFromScale: 12
}],
//Adding tick collection
ticks: [{
type: "majorinterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
},
{
type: "minorinterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
}],
//Adding range collection
ranges: [
{
startWidth: 5, endWidth: 5, startValue: 0, endValue: 200,
backgroundColor: "#94C361", border: {
color: "#94C361", width: 1
}
},
{
startWidth: 5, endWidth: 5, startValue: 200, endValue: 250,
backgroundColor: "#F9CF67", border: {
color: "#F9CF67", width: 1
}
},
{
startWidth: 5, endWidth: 5, startValue: 250, endValue: 300,
backgroundColor: "#F89B83", border: {
color: "#F89B83", width: 1
}
}
],
//Adding indicator collection
indicators: [
{
type: "text", textLocation: { x: 50, y: 100 },
//Adding font option
font: { size: "12px", fontFamily: "Arial", fontType: "Bold" },
//Adding state ranges collection
stateRanges: [{
startValue: 0, endValue: 200,
text: "Safe", textColor: "#94C361"
},
{
startValue: 200, endValue: 250,
text: "Caution", textColor: "#F9CF67"
},
{
startValue: 250, endValue: 300,
text: "Danger", textColor: "#F89B83"
}
]
}]
```

```
}]  
});  
});
```

Execute the above code to render the following output.



### Multiple Indicator

You can set multiple indicators in a single **Linear Gauge** by adding an array of indicator objects. Refer the following code example for multiple indicator functionality.

#### HTML

```
<div id="LinearGauge1"></div>
```

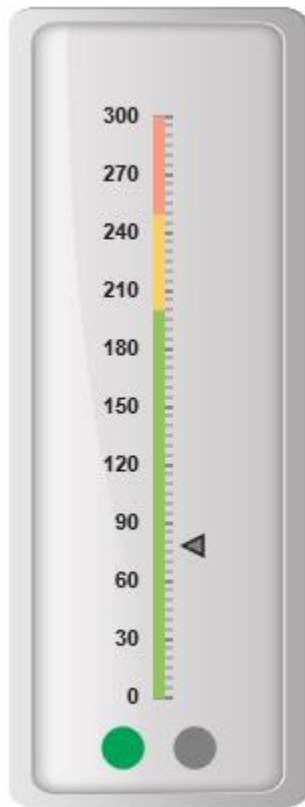
#### JAVASCRIPT

```
$(function () {  
    // For Linear Gauge rendering  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        enableAnimation: false, readOnly: false,  
        //Adding frame object  
        frame: { backgroundImageUrl: "../images/gauge/Gauge_linear_light.png" },  
        value: 78,  
        //Adding scale collection  
        scales: [{  
            width: 0,  
            border: { color: "transparent", width: 0 },  
            minimum: 0,
```

```
maximum: 300,
minorIntervalValue: 5,
majorIntervalValue: 30,
showRanges: true,
showBarPointers: false,
showIndicators: true,
//Adding marker pointer collection
markerPointers: [{
width: 10, length: 10, backgroundColor: "Grey",
distanceFromScale: 12
}],
//Adding ticks collection
ticks: [{
type: "majorinterval", width: 2,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
},
{
type: "minorinterval", width: 1, height: 6,
color: "#8c8c8c", distanceFromScale: { x: 7, y: 0 }
}],
//Adding ranges collection
ranges: [
{
startWidth: 5, endWidth: 5, startValue: 0, endValue: 200,
backgroundColor: "#94C361", border: {
color: "#94C361", width: 1
}
},
{
startWidth: 5, endWidth: 5, startValue: 200, endValue: 250,
backgroundColor: "#F9CF67", border: {
color: "#F9CF67", width: 1
}
},
{
startWidth: 5, endWidth: 5, startValue: 250, endValue: 300,
backgroundColor: "#F89B83", border: {
color: "#F89B83", width: 1
}
}
],
//Adding indicator collection
indicators: [
//Adding indicator 1
{
height: 10, width: 10, type: "circle", position: { x: 30, y: 100 },
//Adding state ranges collection
stateRanges: [{
backgroundColor: "#02A258", endValue: 200,
startValue: 0, borderColor: "#02A258"
},
{
backgroundColor: "Grey", endValue: 300,
startValue: 200, borderColor: "Grey"
}]
},
//Adding indicator 2
```

```
{
  height: 10, width: 10, type: "circle", position: { x: 70, y: 100 },
  stateRanges: [{
    backgroundColor: "Grey", endValue: 200,
    startValue: 0, borderColor: "Grey"
  },
  {
    backgroundColor: "Red", endValue: 300,
    startValue: 200, borderColor: "Red"
  }]
}]
});
```

Execute the above code to render the following output.



## Exporting

**Linear Gauge** has an exporting feature that converts **Gauge** control into image format and then export in client side. The method API `exportImage` is used to export the **LinearGauge**. It has two arguments such as **file name** and **file format** to specify the file name and file formats. For exporting refer the following code example.

### HTML

```
<div id="LinearGauge1"></div>
<button id="btnSubmit">Export</button>
<div id=" fileName ">FileName </div>
```

```
<div id="fileFormat">FileFormat </div>
<select id="fileFormat">
<option value="JPEG">JPEG</option>
<option value="PNG">PNG</option>
</select>
```

## JAVASCRIPT

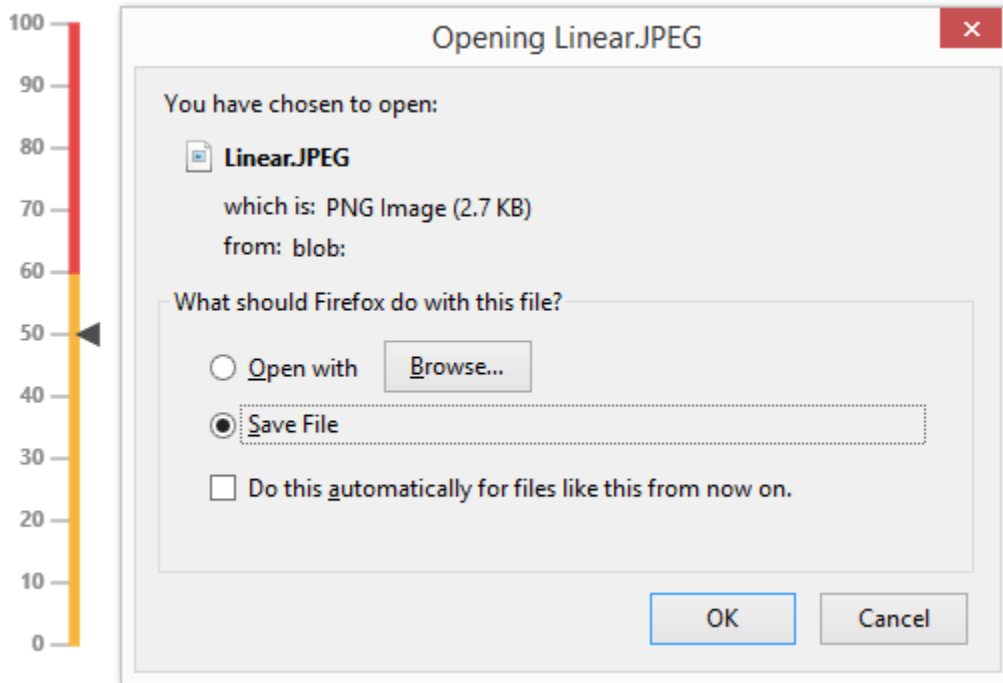
```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module LinearGaugeComponent {
$(function () {
// declaration
var basicButton = new ej.Button($("#btnSubmit"), {
width: "50px", click: "buttonClickEvent",
});
var basicButton = new ej.DopDownList($("#fileFormat"), {
selectedItemIndex: 0, width: "115"
});
//For rendering linear gauge
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
labelColor: "#8c8c8c", width: 450, load: "loadGaugeTheme",
//Adding scale collection
scales: [{
width: 4, border: { color: "transparent", width: 0 }, showBarPointers:
false, showRanges: true, length: 310,
position: { x: 52, y: 50 }, markerPointers: [{
value: 50, length: 10, width: 10, backgroundColor: "#4D4D4D", border: {
color: "#4D4D4D" }
}],
//Adding label collection
labels: [{ font: { size: "11px", fontFamily: "Segoe UI", fontStyle: "bold"
}, distanceFromScale: { x: -13 } }],
//Adding tick collection
ticks: [{ type: "majorinterval", width: 1, color: "#8c8c8c" }],
//Adding range collection
ranges: [{
endValue: 60,
startValue: 0,
backgroundColor: "#F6B53F",
border: { color: "#F6B53F" }, startWidth: 4, endWidth: 4
}, {
endValue: 100,
startValue: 60,
backgroundColor: "#E94649",
border: { color: "#E94649" }, startWidth: 4, endWidth: 4
}
]
}
});
});
}
function buttonClickEvent() {
var FileName = $("#fileName").val();
var FileFormat = new ej.DopDownList($("#fileFormat"));
FileFormat.option(value);
```

```

var linearGaugeSample = new
ej.datavisualization.LinearGauge($("#lineargauge"));
linearGaugeSample.exportImage(FileName,FileFormat);
}
$("#sampleProperties").ejPropertiesPanel();

```

Execute the above code to render the following output.



## MVVM

### AngularJS

**Linear Gauge** contains AngularJS support. It is possible to add object as well as array object in the **Linear Gauge**. The two way binding support is given to the pointer **value**, **minimum** scale value and **maximum** scale value.

### Rendering the Linear gauge

**ej-LinearGauge** is the control tag, where ej is tag prefix and **LinearGauge** is the control name.

**Linear Gauge** is rendered with the following code example.

### HTML

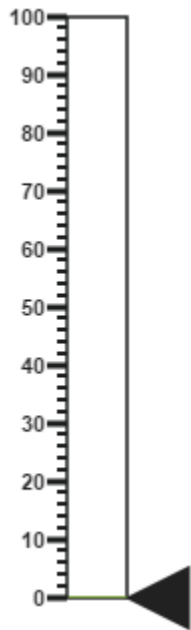
```

<!--To Render the Linear gauge-->
<!doctype html>
<html ng-app="syncApp">
<head>
<!--Refer the necessary script here-->
</head>
<body ng-controller="LinearGauge">
<ej-lineargauge id="linearCore" e-readonly="false" e-load="loadGaugeTheme"
e-enableanimation="false" e-labelcolor="#8c8c8c"></ej-lineargauge>
<script type="text/javascript">
<!--binding the value to the scope variables in application controller-->

```

```
angular.module('syncApp', ['ejangular'])
.controller('LinearGauge', function ($scope) {
$scope.number = 0;
});
</script>
</body>
</html>
```

Execute the above code to render the following output.



#### Adding Scale collection

Scale is an array object and the inner tag is used for it. You can extend the Object in the array collection such as, position with hyphen in the same tag.

**Example:** e-position-x and e-position-y.

#### HTML

```
<!--To Render the Linear gauge-->
<ej-linear gauge id="linearCore">
<!--Adding Scale collection to the Linear gauge-->
<e-scales>
<e-scale e-width="4" e-border-color="transparent" e-border-width="0"
e-showBarPointers="false" e-showRanges="true" e-length="310"
e-position-x="52" e-position-y="50" e-maximum="120"></e-scale>
</e-scales>
</ej-linear gauge>
```

Execute the above code to render the following output.



### Adding Marker Pointer collection

Marker Pointer is an array object and the inner tag is used for it. You can extend the Object in the array collection such as, border with hyphen in the same tag.

**Example:** e-border-color.

### HTML

```
<!--To Render the Linear gauge-->
<ej-lineargauge id="linearCore">
  <!--Adding Scale collection to the Linear gauge-->
  <e-scales>
    <e-scale>
      <!--Adding marker pointer collection to the Scale collection-->
      <e-markerPointers>
        <e-markerPointer e-length="10" e-width="10" e-value="50"
        e-backgroundColor="#4D4D4D"
        e-border-color="#4D4D4D"></e-markerPointer>
      </e-markerPointers>
    </e-scale>
  </e-scales>
</ej-lineargauge>
```

Execute the above code to render the following output.





### Adding label collection

Label is also an array object and the inner tag is used for it. You can extend the Object in the array collection such as, font with hyphen in the same tag.

**Example:** e-font-size.

### HTML

```
<!--To Render the Linear gauge-->
<ej-linear gauge id="linearCore">
  <!--Adding Scale collection to the Linear gauge-->
  <e-scales>
    <e-scale>
      <!--Adding marker pointer collection to the Scale collection-->
      <e-markerPointers>...</e-markerPointers>
      <!--Adding label collection to the Scale collection-->
      <e-labels>
        <e-label e-distanceFromScale-x="-10" e-distanceFromScale-y="0"
        e-font-family="Segoe UI" e-font-fontStyle="bold"
        e-font-size="11px"></e-label>
      </e-labels>
    </e-scale>
  </e-scales>
</ej-linear gauge>
```

Execute the above code to render the following output.



### Adding Tick collection

Tick is an array object and the inner tag is used for it.

### HTML

```
<!--To Render the Linear gauge-->
<ej-lineargauge id="linearCore">
  <!--Adding Scale collection to the Linear gauge-->
  <e-scales>
    <e-scale>
      <!--Adding marker pointer collection to the Scale collection-->
      <e-markerPointers>...</e-markerPointers>
      <!--Adding label collection to the Scale collection-->
      <e-labels>...</e-labels>
      <!--Adding tick collection to the Scale collection-->
      <e-ticks>
        <e-tick e-type="majorinterval" e-width="2" e-color="#8c8c8c"></e-tick>
      </e-ticks>
    </e-scale>
  </e-scales>
</ej-lineargauge>
```

Execute the above code to render the following output.

120—  
 110—  
 100—  
 90—  
 80—  
 70—  
 60—  
 50—  
 40—  
 30—  
 20—  
 10—  
 0—

### Adding Range collection

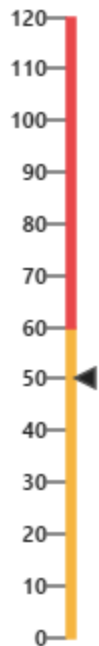
Range is an array object and the inner tag is used for it. You can extend the Object in the array collection such as, border with hyphen in the same tag.

**Example:** e-border-color.

### HTML

```
<!--To Render the Linear gauge-->
<ej-lineargauge id="linearCore">
  <!--Adding Scale collection to the Linear gauge-->
  <e-scales>
    <e-scale>
      <!--Adding marker pointer collection to the Scale collection-->
      <e-markerPointers>...</e-markerPointers>
      <!--Adding label collection to the Scale collection-->
      <e-labels>...</e-labels>
      <!--Adding tick collection to the Scale collection-->
      <e-ticks>...</e-ticks>
      <!--Adding range collection to the Scale collection-->
      <e-ranges>
        <e-range e-startValue="0" e-endValue="60" e-startWidth="4"
        e-endWidth="4" e-backgroundColor="#F6B53F"
        e-border-color="#F6B53F"></e-range>
        <e-range e-startValue="60" e-endValue="120" e-startWidth="4"
        e-endWidth="4" e-backgroundColor="#E94649"
        e-border-color="#E94649"></e-range>
      </e-ranges>
    </e-scale>
  </e-scales>
</ej-lineargauge>
```

Finally while running the above codes, the resultant gauge appears as follows.



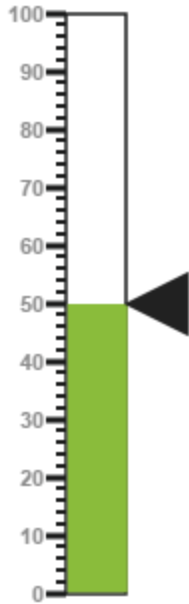
### Two Way Binding

**Linear Gauge** support the two way binding for the property **value**, **minimum** and **maximum** as mentioned earlier. Following code example explains how to achieve the two way binding to the **Linear Gauge**.

### HTML

```
<!doctype html>
<html ng-app="syncApp">
<head>
<!--Refer the necessary script here-->
</head>
<body ng-controller="LinearGauge">
<div id="linearframe">
<ej-lineargauge id="linearCore" e-value="number" e-readonly="false" e-
load="loadGaugeTheme" e-enableanimation="false" e-labelcolor="#8c8c8c">
</ej-lineargauge>
</div>
<input type="text" id="txtMax" e-value="number" ej-numerictextbox ng-
model="number" e-decimalplaces="2" e-showspinbutton="false" style="width:
110px" />
<script type="text/javascript">
<!--binding the value to the scope variables in application controller-->
angular.module('syncApp', ['ejangular'])
.controller('LinearGauge', function ($scope) {
$scope.number = 50;
});
</script>
</body>
</html>
```

Execute the above code to render the following output.



## Events

### *drawBarPointers*

Triggers while the bar pointer are being drawn on the gauge, you can use `drawBarPointers` event.

### **JAVASCRIPT**

```
<script>
//drawBarPointers event for linear gauge
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#gauge"), {
drawBarPointers: function () {
//...//
}
});
});
</script>
```

### *drawCustomLabel*

Triggers while the customLabel are being drawn on the gauge, you can use `drawCustomLabel` event.

### **JAVASCRIPT**

```
<script>
//drawCustomLabel event for linear gauge
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#gauge"), {
drawCustomLabel: function () {
//...//
}
});
});
</script>
```

### *drawIndicators*

Triggers while the Indicator are being drawn on the gauge, you can use `drawIndicators` event.

#### **JAVASCRIPT**

```
<script>
//drawIndicators event for linear gauge
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#gauge"), {
drawIndicators: function () {
//...//
}
});
});
</script>
```

### *drawLabels*

Triggers while the label are being drawn on the gauge, you can use `drawLabels` event.

#### **JAVASCRIPT**

```
<script>
//drawLabels event for linear gauge
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#gauge"), {
drawLabels: function () {
//...//
}
});
});
</script>
```

### *drawMarkerPointers*

Triggers while the marker are being drawn on the gauge, you can use `drawMarkerPointers` event.

#### **JAVASCRIPT**

```
<script>
//drawMarkerPointers event for linear gauge
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#gauge"), {
drawMarkerPointers: function () {
//...//
}
});
});
</script>
```

### *drawRange*

Triggers while the range are being drawn on the gauge, you can use `drawRange` event.

#### **JAVASCRIPT**

```
<script>
//drawRange event for linear gauge
```

```
$(function () {  
  var sample = new ej.datavisualization.LinearGauge($("#gauge"), {  
    drawRange: function () {  
      //...//  
    }  
  });  
});  
</script>
```

### drawTicks

Triggers while the ticks are being drawn on the gauge, you can use **drawTicks** event.

#### JAVASCRIPT

```
<script>  
  //drawTicks event for linear gauge  
  $(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#gauge"), {  
      drawTicks: function () {  
        //...//  
      }  
    });  
  });  
</script>
```

### init

Triggers when the gauge is initialized, you can use **initevent**.

#### JAVASCRIPT

```
<script>  
  //init event for linear gauge  
  $(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#gauge"), {  
      init: function () {  
        //...//  
      }  
    });  
  });  
</script>
```

### load

Triggers while the gauge start to Load, you can use **load** event.

#### JAVASCRIPT

```
<script>  
  //load event for linear gauge  
  $(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#gauge"), {  
      load: function () {  
        //...//  
      }  
    });  
  });  
</script>
```

```
</script>
```

### *mouseClick*

Triggers when the left mouse button is clicked, you can use `mouseClick` event.

#### **JAVASCRIPT**

```
<script>
//mouseClick event for linear gauge
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#gauge"), {
mouseClick: function () {
//...//
}
});
});
</script>
```

### *mouseClickMove*

Triggers when clicking and dragging the mouse pointer over the gauge pointer, you can use `mouseClickMove` event.

#### **JAVASCRIPT**

```
<script>
//mouseClickMove event for linear gauge
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#gauge"), {
mouseClickMove: function () {
//...//
}
});
});
</script>
```

### *mouseClickUp*

Triggers when the mouse click is released, you can use `mouseClickUp` event.

#### **JAVASCRIPT**

```
<script>
//mouseClickUp event for linear gauge
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#gauge"), {
mouseClickUp: function () {
//...//
}
});
});
</script>
```

### *renderComplete*

Triggers while the rendering of the gauge completed, you can use `renderComplete` event.



## JAVASCRIPT

```
<script>
//renderComplete event for linear gauge
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#gauge"), {
renderComplete: function () {
//...//
}
});
});
</script>
```

## Methods

### [destroy\(\)](#)

Destroy the linear gauge all events bound using this.\_on will be unbind automatically and bring the control to pre-init state.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
//...//
});
sample.destroy();
});
});
}
```

### [getBarDistanceFromScale\(\)](#)

To get bar distance from scale in number, you can use `getBarDistanceFromScale` method.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
//...//
});
sample.getBarDistanceFromScale();
});
});
```

```
});  
}
```

### [getBarPointerValue\(\)](#)

To get Bar Pointer Value in number, you can use `getBarPointerValue` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
  $(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
      ///...///  
    });  
    sample.getBarPointerValue();  
  });  
};
```

### [getBarWidth\(\)](#)

To get Bar Width in number, you can use `getBarWidth` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
  $(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
      ///...///  
    });  
    sample.getBarWidth();  
  });  
};
```

### [getCustomLabelAngle\(\)](#)

To get CustomLabel Angle in number, you can use `getCustomLabelAngle` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
///...///
});
sample.getCustomLabelAngle();
});
});
}

```

#### getCustomLabelValue()

To get CustomLabel Value in string, you can use `getCustomLabelValue` method.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
///...///
});
sample.getCustomLabelValue();
});
});
}

```

#### getLabelAngle()

To get Label Angle in number, you can use `getLabelAngle` method.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
///...///
});
sample.getLabelAngle();
});
});
}

```

### [getLabelPlacement\(\)](#)

To get LabelPlacement in number, you can use `getLabelPlacement` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            //...//
        });
        sample.getLabelPlacement();
    });
}
```

### [getLabelStyle\(\)](#)

To get LabelStyle in number, you can use `getLabelStyle` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            //...//
        });
        sample.getLabelStyle();
    });
}
```

### [getLabelXDistanceFromScale\(\)](#)

To get Label XDistance From Scale in number, you can use `getLabelXDistanceFromScale` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
```

```
$(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        //...//  
    });  
    sample.getLabelXDistanceFromScale();  
});  
});  
}
```

#### [getLabelYDistanceFromScale\(\)](#)

To get PointerValue in number, you can use `getLabelXDistanceFromScale` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            //...//  
        });  
        sample.getLabelYDistanceFromScale();  
    });  
});  
}
```

#### [getMajorIntervalValue\(\)](#)

To get Major Interval Value in number, you can use `getMajorIntervalValue` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            //...//  
        });  
        sample.getMajorIntervalValue();  
    });  
});  
}
```

#### [getMarkerStyle\(\)](#)

To get MarkerStyle in number, you can use `getMarkerStyle` method.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
///...///
});
sample.getMarkerStyle();
});
});
}
```

### *getMaximumValue()*

To get Maximum Value in number, you can use `getMaximumValue` method.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
///...///
});
sample.getMaximumValue();
});
});
}
```

### *getMinimumValue()*

To get PointerValue in number, you can use `getMinimumValue` method.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
///...///
});
});
});
}
```

```
});  
sample.getMinimumValue();  
});  
});  
}
```

### *getMinorIntervalValue()*

To get Minor Interval Value in number, you can use `getMinorIntervalValue` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            ///...///  
        });  
        sample.getMinorIntervalValue();  
    });  
});  
}
```

### *getPointerDistanceFromScale()*

To get Pointer Distance From Scale in number, you can use `getPointerDistanceFromScale` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            ///...///  
        });  
        sample.getPointerDistanceFromScale();  
    });  
});  
}
```

### *getPointerHeight()*

To get PointerHeight in number, you can use `getPointerHeight` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      ///...///
    });
    sample.getPointerHeight();
  });
}
```

*getPointerPlacement()*

To get Pointer Placement in String, you can use `getPointerPlacement` method.

**HTML**

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      ///...///
    });
    sample.getPointerPlacement();
  });
}
```

*getPointerValue()*

To get PointerValue in number, you can use `getPointerValue` method.

**HTML**

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      ///...///
    });
    sample.getPointerValue();
  });
}
```



```
});  
}
```

### [getPointerWidth\(\)](#)

To get PointerWidth in number, you can use `getPointerWidth` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
  $(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
      ///...///  
    });  
    sample.getPointerWidth();  
  });  
};
```

### [getRangeBorderWidth\(\)](#)

To get Range Border Width in number, you can use `getRangeBorderWidth` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
  $(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
      ///...///  
    });  
    sample.getRangeBorderWidth();  
  });  
};
```

### [getRangeDistanceFromScale\(\)](#)

To get Range Distance From Scale in number, you can use `getRangeDistanceFromScale` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.getRangeDistanceFromScale();
  });
};
```

### [getRangeEndValue\(\)](#)

To get Range End Value in number, you can use `getRangeEndValue` method.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.getRangeEndValue();
  });
};
```

### [getRangeEndWidth\(\)](#)

To get Range End Width in number, you can use `getRangeEndWidth` method.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.getRangeEndWidth();
  });
};
```

### *getRangePosition()*

To get Range Position in number, you can use `getRangePosition` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            //...//
        });
        sample.getRangePosition();
    });
}
```

### *getRangeStartValue()*

To get Range Start Value in number, you can use `getRangeStartValue` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            //...//
        });
        sample.getRangeStartValue();
    });
}
```

### *getRangeStartWidth()*

To get Range Start Width in number, you can use `getRangeStartWidth` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
```

```
$(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        //...//  
    });  
    sample.getRangeStartWidth();  
});  
});  
}
```

#### [getScaleBarLength\(\)](#)

To get ScaleBarLength in number, you can use `getScaleBarLength` method.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            //...//  
        });  
        sample.getScaleBarLength();  
    });  
});  
}
```

#### [getScaleBarSize\(\)](#)

To get Scale Bar Size in number, you can use `getScaleBarSize` method.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            //...//  
        });  
        sample.getScaleBarSize();  
    });  
});  
}
```

#### [getScaleBorderWidth\(\)](#)

To get Scale Border Width in number, you can use `getScaleBorderWidth` method.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            ///...///
        });
        sample.getScaleBorderWidth();
    });
};
```

### [getScaleDirection\(\)](#)

To get Scale Direction in number, you can use `getScaleDirection` method.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            ///...///
        });
        sample.getScaleDirection();
    });
};
```

### [getScaleLocation\(\)](#)

To get Scale Location in object, you can use `getScaleLocation` method.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            ///...///
        });
    });
};
```

```
});  
sample.getScaleLocation();  
});  
});  
}
```

### *getScaleStyle()*

To get Scale Style in string, you can use `getScaleStyle` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            ///...  
        });  
        sample.getScaleStyle();  
    });  
});  
}
```

### *getTickAngle()*

To get Tick Angle in number, you can use `getTickAngle` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            ///...  
        });  
        sample.getTickAngle();  
    });  
});  
}
```

### *getTickHeight()*

To get Tick Height in number, you can use `getTickHeight` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      ///...///
    });
    sample.getTickHeight();
  });
};
```

*getTickPlacement()*

To get getTickPlacement in number, you can use `getTickPlacement` method.

**HTML**

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      ///...///
    });
    sample.getTickPlacement();
  });
};
```

*getTickStyle()*

To get Tick Style in string, you can use `getTickStyle` method.

**HTML**

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      ///...///
    });
    sample.getTickStyle();
  });
};
```

```
});  
}
```

### [getTickWidth\(\)](#)

To get Tick Width in number, you can use `getTickWidth` method.

### HTML

```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
  $(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
      ///...  
    });  
    sample.getTickWidth();  
  });  
};
```

### [getTickXDistanceFromScale\(\)](#)

To get Tick XDistance From Scale in number, you can use `getTickXDistanceFromScale` method.

### HTML

```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
  $(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
      ///...  
    });  
    sample.getTickXDistanceFromScale();  
  });  
};
```

### [getTickYDistanceFromScale\(\)](#)

To get Tick YDistance From Scale in number, you can use `getTickYDistanceFromScale` method.

### HTML

```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT



```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.getTickYDistanceFromScale();
  });
};
```

### *scales()*

Specifies the scales, you can use `scales` method.

### HTML

```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.scales();
  });
};
```

### *setBarDistanceFromScale()*

To set `setBarDistanceFromScale`, you can use `setBarDistanceFromScale` method.

### HTML

```
<div id="LinearGauge1"></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.setBarDistanceFromScale();
  });
};
```

### setBarPointerValue()

To set setBarPointerValue, you can use `setBarPointerValue` method.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            //...//
        });
        sample.setBarPointerValue();
    });
}
```

### setBarWidth()

To set setBarWidth, you can use `setBarWidth` method.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            //...//
        });
        sample.setBarWidth();
    });
}
```

### setCustomLabelAngle()

To set setCustomLabelAngle, you can use `setCustomLabelAngle` method.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
```

```
$(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        //...//  
    });  
    sample.setCustomLabelAngle();  
});  
});  
}
```

#### [setCustomLabelValue\(\)](#)

To set setCustomLabelValue, you can use `setCustomLabelValue` method.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            //...//  
        });  
        sample.setCustomLabelValue();  
    });  
});  
}
```

#### [setLabelAngle\(\)](#)

To set setLabelAngle, you can use `setLabelAngle` method.

#### HTML

```
<div id="LinearGauge1"></div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            //...//  
        });  
        sample.setLabelAngle();  
    });  
});  
}
```

#### [setLabelPlacement\(\)](#)

To set setLabelPlacement, you can use `setLabelPlacement` method.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
///...///
});
sample.setLabelPlacement();
});
});
}
```

### [setLabelStyle\(\)](#)

To set setLabelStyle, you can use `setLabelStyle` method.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
///...///
});
sample.setLabelStyle();
});
});
}
```

### [setLabelXDistanceFromScale\(\)](#)

To set setLabelXDistanceFromScale, you can use `setLabelXDistanceFromScale` method.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
$(function () {
var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
///...///
});
});
});
}
```

```
});  
sample.setLabelXDistanceFromScale();  
});  
});  
}
```

#### *setLabelYDistanceFromScale()*

To set setLabelYDistanceFromScale, you can use `setLabelYDistanceFromScale` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            ///...  
        });  
        sample.setLabelYDistanceFromScale();  
    });  
});  
}
```

#### *setMajorIntervalValue()*

To set setMajorIntervalValue, you can use `setMajorIntervalValue` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            ///...  
        });  
        sample.setMajorIntervalValue();  
    });  
});  
}
```

#### *setMarkerStyle()*

To set setMarkerStyle, you can use `setMarkerStyle` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      ///...///
    });
    sample.setMarkerStyle();
  });
}
```

*setMaximumValue()*

To set setMaximumValue, you can use **setMaximumValue** method.

**HTML**

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      ///...///
    });
    sample.setMaximumValue();
  });
}
```

*setMinimumValue()*

To set setMinimumValue, you can use **setMinimumValue** method.

**HTML**

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      ///...///
    });
    sample.setMinimumValue();
  });
}
```

```
});  
}
```

### *setMinorIntervalValue()*

To set setMinorIntervalValue, you can use `setMinorIntervalValue` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
  $(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
      ///...  
    });  
    sample.setMinorIntervalValue();  
  });  
};
```

### *setPointerDistanceFromScale()*

To set setPointerDistanceFromScale, you can use `setPointerDistanceFromScale` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
  $(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
      ///...  
    });  
    sample.setPointerDistanceFromScale();  
  });  
};
```

### *setPointerHeight()*

To set PointerHeight, you can use `setPointerHeight` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.setPointerHeight();
  });
};
```

#### *setPointerPlacement()*

To set setPointerPlacement, you can use **setPointerPlacement** method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.setPointerPlacement();
  });
};
```

#### *setPointerValue()*

To set PointerValue, you can use **setPointerValue** method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.setPointerValue();
  });
};
```



### *setPointerWidth()*

To set PointerWidth, you can use `setPointerWidth` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            ///...///
        });
        sample.setPointerWidth();
    });
}
```

### *setRangeBorderWidth()*

To set setRangeBorderWidth, you can use `setRangeBorderWidth` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            ///...///
        });
        sample.setRangeBorderWidth();
    });
}
```

### *setRangeDistanceFromScale()*

To set setRangeDistanceFromScale, you can use `setRangeDistanceFromScale` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
```

```
$(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
        //...  
    });  
    sample.setRangeDistanceFromScale();  
});  
});  
}
```

#### *setRangeEndValue()*

To set setRangeEndValue, you can use `setRangeEndValue` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            //...  
        });  
        sample.setRangeEndValue();  
    });  
});  
}
```

#### *setRangeEndWidth()*

To set setRangeEndWidth, you can use `setRangeEndWidth` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            //...  
        });  
        sample.setRangeEndWidth();  
    });  
});  
}
```

#### *setRangePosition()*

To set setRangePosition, you can use `setRangePosition` method.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            //...//
        });
        sample.setRangePosition();
    });
}
```

### setRangeStartValue()

To set setRangeStartValue, you can use `setRangeStartValue` method.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            //...//
        });
        sample.setRangeStartValue();
    });
}
```

### setRangeStartWidth()

To set setRangeStartWidth, you can use `setRangeStartWidth` method.

## HTML

```
<div id="LinearGauge1"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            //...//
        });
    });
}
```

```
});  
sample.setRangeStartWidth();  
});  
});  
}
```

### *setScaleBarLength()*

To set `setScaleBarLength`, you can use `setScaleBarLength` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            ///...  
        });  
        sample.setScaleBarLength();  
    });  
};
```

### *setScaleBarSize()*

To set `setScaleBarSize`, you can use `setScaleBarSize` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
    $(function () {  
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
            ///...  
        });  
        sample.setScaleBarSize();  
    });  
};
```

### *setScaleBorderWidth()*

To set `setScaleBorderWidth`, you can use `setScaleBorderWidth` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.setScaleBorderWidth();
  });
};
```

*setScaleDirection()*

To set setScaleDirection, you can use `setScaleDirection` method.

**HTML**

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.setScaleDirection();
  });
};
```

*setScaleLocation()*

To set setScaleLocation, you can use `setScaleLocation` method.

**HTML**

```
<div id="LinearGauge1"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.setScaleLocation();
  });
};
```

```
});  
}
```

### *setScaleStyle()*

To set setScaleStyle, you can use `setScaleStyle` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
  $(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
      ///...  
    });  
    sample.setScaleStyle();  
  });  
};
```

### *setTickAngle()*

To set setTickAngle, you can use `setTickAngle` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module LinearGaugeComponent {  
  $(function () {  
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {  
      ///...  
    });  
    sample.setTickAngle();  
  });  
};
```

### *setTickHeight()*

To set setTickHeight, you can use `setTickHeight` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.setTickHeight();
  });
};
```

#### *setTickPlacement()*

To set setTickPlacement, you can use `setTickPlacement` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.setTickPlacement();
  });
};
```

#### *setTickStyle()*

To set setTickStyle, you can use `setTickStyle` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
  $(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
      //...//
    });
    sample.setTickStyle();
  });
};
```

### *setTickWidth()*

To set setTickWidth, you can use `setTickWidth` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            //...//
        });
        sample.setTickWidth();
    });
}
```

### *setTickXDistanceFromScale()*

To set setTickXDistanceFromScale, you can use `setTickXDistanceFromScale` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
    $(function () {
        var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
            //...//
        });
        sample.setTickXDistanceFromScale();
    });
}
```

### *setTickYDistanceFromScale()*

To set setTickYDistanceFromScale, you can use `setTickYDistanceFromScale` method.

#### **HTML**

```
<div id="LinearGauge1"></div>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
```



```
$(function () {
    var sample = new ej.datavisualization.LinearGauge($("#LinearGauge1"), {
        //...//
    });
    sample.setTickYDistanceFromScale();
});
});
}
```

## ListBox

### Overview

The Essential **Typescript** ListBox control provides a list of options for users to select from the lists. It can include other elements such as images, text boxes, check boxes and so on. The following are the key features of the Essential **Typescript** ListBox control.

#### Key Features

**Data Binding:** Supports Data binding with JSON data and remote data.

**Multi Selection:** Supports multiple selection of list items.

**Virtual Scrolling:** Provides support to load its data on demand through virtual scrolling which greatly improves the application's performance.

**Template Support:** Support Template contents to render as list items

**Grouping:** Groups the set of list items with header

**Cascading:** To populate data in a ListBox based on the selection in another ListBox.

**Drag and Drop:** Supports drag and drop features for list items.

### Getting Started

Using the following steps, you can create a **Typescript** ListBox component. The basic rendering of **Typescript** ListBox is achieved with default functionality.

#### Create a ListBox

You can create a **Typescript** application with the help of the given [Typescript Getting Started Documentation](#).

Within an index.html file and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Typescript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.web
.all.min.js" type="text/javascript"></script>
```

```
</head>
<body>
<!--Add ListBox here-->
</body>
</html>
```

Create UL and LI elements and add in the <body> tag. To create the ListBox, you should call the `ejListBox` jQuery plug-in function.

### HTML

```
<div>
<ul id="listbox">
<li>Audi A4</li>
<li>Audi A5</li>
<li>Audi A6</li>
<li>Audi A7</li>
<li>Audi A8</li>
<li>BMW 501</li>
<li>BMW 502</li>
<li>BMW 503</li>
<li>Batch</li>
<li>BMW 507</li>
<li>BMW 3200</li>
<li>Cut</li>
</ul>
</div>
<script src="app.js"></script>
```

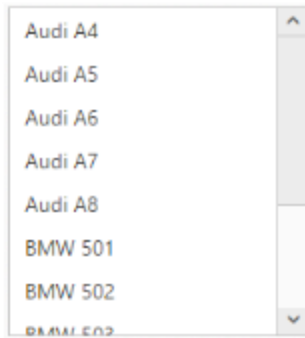
Initialize the ListBox in app.ts file by using the ej.ListBox method.

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ListComponent {
$(function () {
let sample = new ej.ListBox($("#listbox"));
});
}
```

Now build your application, so that the **app.js** file is automatically generated and got added to your project (User have nothing to do with this file). Now, whatever code changes that you make in **app.ts** file will be reflected in app.js file automatically.

Run the above code to render the following output:



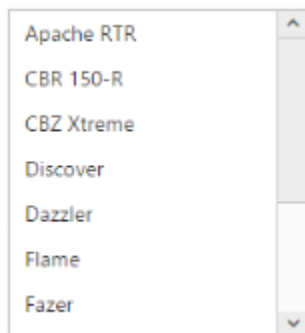
## Data Binding

We can populate data in the ListBox widget using “dataSource” and “fields” properties

### JS

```
<ul id="listbox"></ul>
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ListComponent {
$(function () {
let bikeList = [
{ bikeId: "bk1", bikeName: "Apache RTR" },
{ bikeId: "bk2", bikeName: "CBR 150-R" },
{ bikeId: "bk3", bikeName: "CBZ Xtreme" },
{ bikeId: "bk4", bikeName: "Discover" },
{ bikeId: "bk5", bikeName: "Dazzler" },
{ bikeId: "bk6", bikeName: "Flame" },
{ bikeId: "bk7", bikeName: "Fazer" },
{ bikeId: "bk8", bikeName: "FZ-S" },
{ bikeId: "bk9", bikeName: "Pulsar" },
{ bikeId: "bk10", bikeName: "Shine" },
{ bikeId: "bk11", bikeName: "R15" },
{ bikeId: "bk12", bikeName: "Unicorn" }
];
let sample = new ej.ListBox($("#listbox"), { dataSource: bikeList, fields: {
text: "bikeName", id: "bikeId" } });
});
}
```

Run the above code to render the following output:



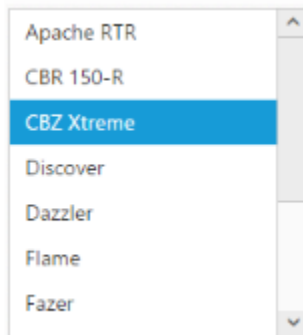
## Selection

The ListBox widget supports the item selection.

### JS

```
let sample = new ej.ListBox($("#listbox"), { dataSource: bikeList,
selectedIndex: 2, fields: { text: "bikeName", id: "bikeId" } });
```

Run the above code to render the following output:



*Note: You can find the ListBox properties from the [API reference](#) document*

## Data binding

### Field mapping

The ListBox widget has a field property (object) which holds the properties to map with datasource fields. For example, the field object has a text property which is necessary to map with specific field in the datasource to render the items in the ListBox widget.

The field object contains the following properties.

- [text](#)
- [toolTipText](#)
- [id](#)
- [selectBy](#)
- [groupBy](#)
- [checkBy](#)
- [tableName](#)
- [imageUrl](#)
- [imageAttributes](#)
- [spriteCssClass](#)
- [htmlAttributes](#)

### Local data

The local data can be an array of JSON objects which is assigned for the ListBox widget's datasource property. Refer the below example.

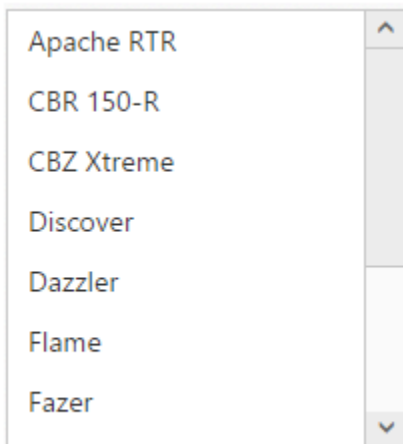
Here the bikeName and bikeId fields are mapped with text and id properties of the field object respectively.

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ListBoxComponent {
$(function () {
bikeList = [{ bikeId: "bk1", bikeName: "Apache RTR" },
{ bikeId: "bk2", bikeName: "CBR 150-R" },
{ bikeId: "bk3", bikeName: "CBZ Xtreme" },
{ bikeId: "bk4", bikeName: "Discover" },
{ bikeId: "bk5", bikeName: "Dazzler" },
{ bikeId: "bk6", bikeName: "Flame" },
{ bikeId: "bk7", bikeName: "Fazer" },
{ bikeId: "bk8", bikeName: "FZ-S" },
{ bikeId: "bk9", bikeName: "Pulsar" },
{ bikeId: "bk10", bikeName: "Shine" },
{ bikeId: "bk11", bikeName: "R15" },
{ bikeId: "bk12", bikeName: "Unicorn" }];
var listBoxInstance = new ej.ListBox($("#listbox"), {
dataSource:bikeList,
fields: { id: "bikeId", text: "bikeName" }
});
});
}

```



## Remote data

### OData

[OData](#) is a standardized protocol for creating and consuming the data. You can retrieve data from OData service by using [ej.DataManager](#).

Here the CustomerID field is mapped with text property of the field object. The queries can be created using [ej.Query\(\)](#).

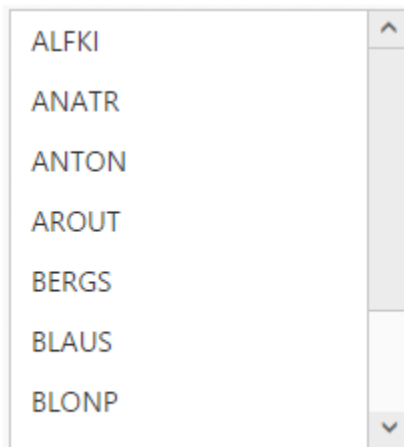
### HTML

```

<ul id="listbox">
</ul>
<script type="text/javascript">
$(function () {
//create data manager
var dataManager = ej.DataManager({

```

```
//OData service
url: "http://mvc.syncfusion.com/Services/Northwnd.svc/"
});
// create query
var query = ej.Query().from("Customers").take(10);
var listBoxInstance = new ej.ListBox($("#listbox"), {
dataSource: dataManager,
fields: { text: "CustomerID" },
query: query
});
});
</script>
```



### WebAPI

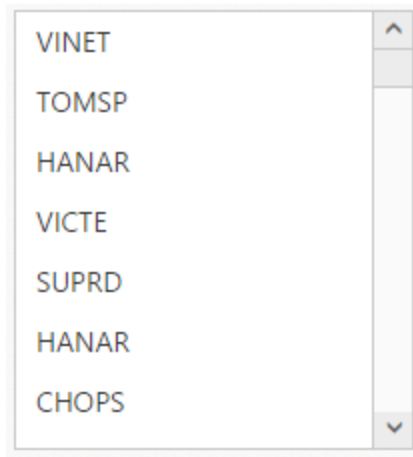
[ASP.NET Web API](#) is a Framework for building HTTP services. You can retrieve data from ASP.NET Web API by using [ej.DataManager](#).

### JAVASCRIPT

```
$(function () {
//create data manager
var dataManager = ej.DataManager({
//ASP.NET Web API
url: "http://mvc.syncfusion.com/UGService/api/Orders",
crossDomain: true
});
var listBoxInstance = new ej.ListBox($("#listbox"), {
dataSource: dataManager,
fields: { text: "CustomerID" }
});
});
```

**Note:** In the above data manager configuration, "crossDomain" must be set to true to access the data from Web API.

### [Cross domain](#)



### Virtual Scrolling

The ListBox widget provides support to load its data on demand via scrolling behavior to improve the application's performance. This can be achieved using `allowVirtualScrolling` property. There are two ways to load data based on the scrolling type.

1. Normal scrolling
2. Continuous Scrolling

The scrolling type can be defined via `virtualScrollMode` property.

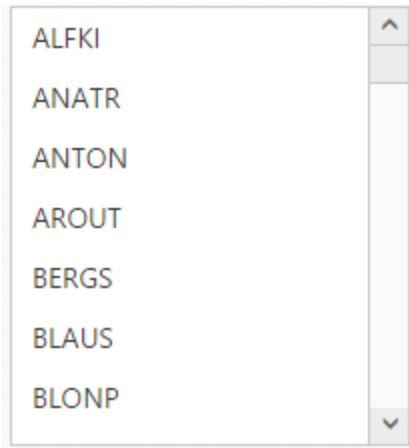
### Normal Scrolling

This mode allows you to load the list box data while scrolling i.e. each time the scroll bar is scrolled, it will send request to the server to load the data.

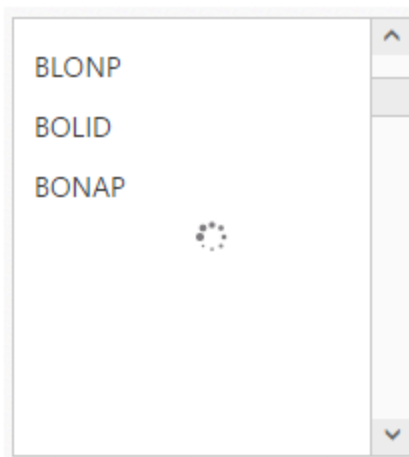
### JAVASCRIPT

```
$(function () {
    //create data manager
    var dataManager = ej.DataManager({
        //OData service
        url: "http://mvc.syncfusion.com/Services/Northwnd.svc/"
    });
    // create query
    var query = ej.Query().from("Customers");
    var listBoxInstance = new ej.ListBox($("#listbox"), {
        dataSource: dataManager,
        fields: { text: "CustomerID" },
        query: query,
        allowVirtualScrolling: true
    });
});
```

**Note:** By default, the value of "virtualScrollMode" property is normal.



*Before scrolling*



*Virtual scrolling (normal)*

### Continuous Scrolling

This mode allows you to load the list box data when the scrollbar reaches the end point. In this mode, we can specify the number of items to be loaded per request.

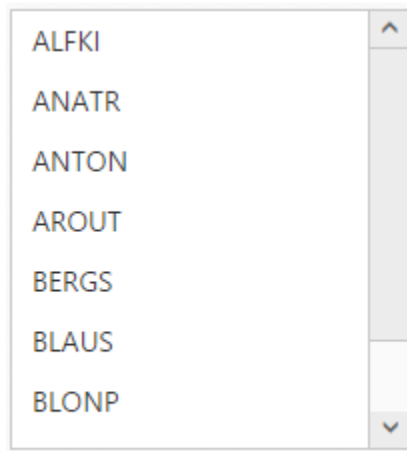
The number of items to be loaded per request can be specified using the “itemRequestCount” property.

### JAVASCRIPT

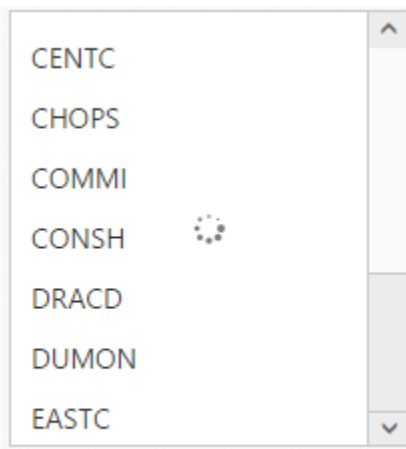
```
$(function () {
    //create data manager
    var dataManager = ej.DataManager({
        //OData service
        url: "http://mvc.syncfusion.com/Services/Northwnd.svc/"
    });
    // create query
    var query = ej.Query().from("Customers");
    var listBoxInstance = new ej.ListBox($("#listbox"), {
        dataSource: dataManager,
        fields: { text: "CustomerID" },
        query: query,
```



```
allowVirtualScrolling: true,
// specifying the scroll mode
virtualScrollMode: ej.VirtualScrollMode.Continuous,
//specifying the number of items to be loaded per request
itemRequestCount: 10
});
});
```



*Before scrolling*



*Virtual scrolling (continuous)*

### Handling errors

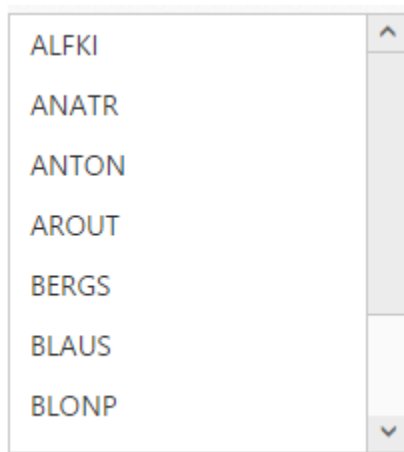
In remote binding, the server might not return data sometimes due to various reasons. In such cases we need to handle the error properly. We can handle it using the “actionFailure” event.

[actionComplete](#) and [actionSuccess](#)

### JAVASCRIPT

```
$(function () {
//create data manager
var dataManager = ej.DataManager({
```

```
//OData service
url: "http://mvc.syncfusion.com/Services/Northwnd.svc/"
});
// create query
var query = ej.Query()
.from("Customers").take(10);
var listBoxInstance = new ej.ListBox($("#listbox"), {
dataSource: dataManager,
fields: { text: "CustomerID" },
query: query,
actionFailure: "onFailure"
});
});
function onFailure(args) {
//handle errors
}
```



## Selection

The ListBox widget allows you to highlight the selected item. It allows multiple selection also.

### Selection on initialize

By default, the ListBox widget allows single item selection. We can select specific item during initialization of the ListBox widget using the “selectedIndex” API.

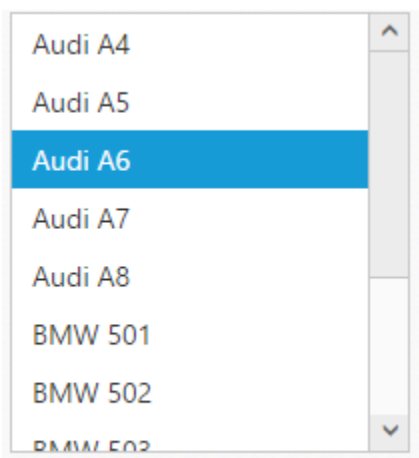
## HTML

```
<div>
<ul id="listbox">
<li>Audi A4</li>
<li>Audi A5</li>
<li>Audi A6</li>
<li>Audi A7</li>
<li>Audi A8</li>
<li>BMW 501</li>
<li>BMW 502</li>
<li>BMW 503</li>
<li>Batch</li>
<li>BMW 507</li>
<li>BMW 3200</li>
<li>Cut</li>
</ul>
```

```
</ul>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ListBoxComponent {
    $(function () {
        var listboxInstance = new ej.ListBox($("#listbox"), {
            selectedIndex: 2
        });
    });
}
```



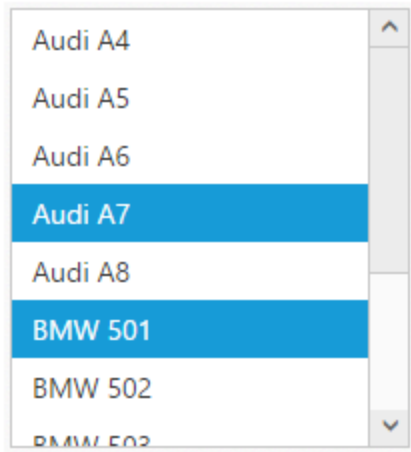
### Multiple selection

Multiple selection can be enabled using “allowMultiSelection” property. You can select multiple list items using “Ctrl” and “Shift” keys.

### [Keyboard Interaction.](#)

### JAVASCRIPT

```
$(function () {
    var listboxInstance = new ej.ListBox($("#listbox"), {
        allowMultiSelection: true
    });
});
```



### Checkbox

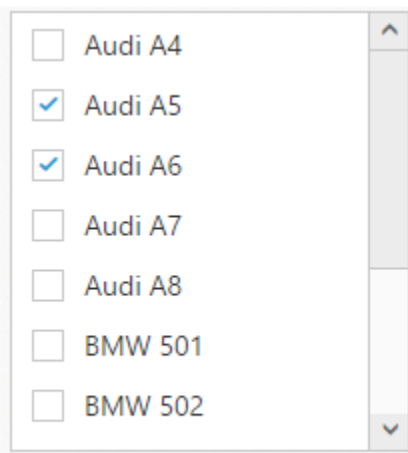
The ListBox widget allows selection through checkbox. It can be enabled using “showCheckbox” API.

The specified items can be checked on initialize through “checkedIndices” property.

[checkedIndices](#).

### JAVASCRIPT

```
$(function () {
    var listboxInstance = new ej.ListBox($("#listbox"), {
        showCheckbox: true,
        checkedIndices: [1, 2]
    });
});
```



### Sorting

We can change ListBox items rendering order either as ascending or descending, by using [ej.SortOrder](#) property. By default ej.SortOrder will be "None". Please use code as like in below,

### HTML

```
<div class="contents">
<ul id="selectsection"></ul>
</div>
```

## JAVASCRIPT

```
<script type="text/javascript">
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ListBoxComponent {
$(function () {
var skills = [{ skill: "F#" }, { skill: "ActionScript" }, { skill: "Delphi"
}, { skill: "Basic" },
{ skill: "C++" }, { skill: "ESPOL" }, { skill: "C#" }, { skill: "DBase" }, {
skill: "ASP.NET" }
];
var listBoxInstance = new ej.ListBox($("#selectsection"), {
width: "220", dataSource: skills,
fields: { text: "skill" },
sortOrder:ej.SortOrder.Descending
})
});
}
</script>
```

## Select items



## Grouping

ListBox items can be grouped by providing a heading (header) for each set of items. It can be done in two ways.

1. Using span tag
2. Data binding

### Using span tag

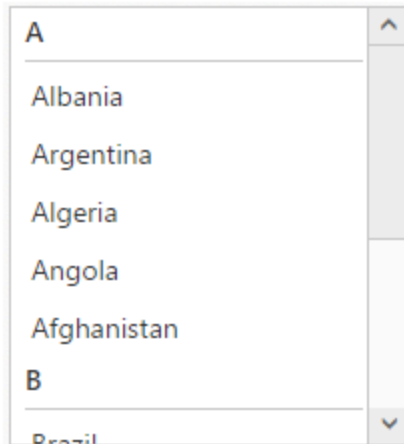
The header for each group can be defined using the “span” element”.

#### HTML

```
<!--grouped listbox-->
<ul id="listbox">
  <!--header-->
  <span class="e-ghead">A</span>
  <li>Albania</li>
  <li>Argentina</li>
  <li>Algeria</li>
  <li>Angola</li>
  <li>Afghanistan</li>
  <!--header-->
  <span class="e-ghead">B</span>
  <li>Brazil</li>
  <li>Bahrain</li>
  <li>Burma</li>
  <li>Barbados</li>
  <li>Botswana</li>
  <li>Belarus</li>
  <li>Bolivia</li>
</ul>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ListBoxComponent {
  $(function () {
    var listboxInstance = new ej.ListBox($("#listbox"), {
    });
  });
}
```



### Data binding

The grouped ListBox can be also created via data binding which is explained below. The data items can be categorized by using a specific field in the ListBox widget.

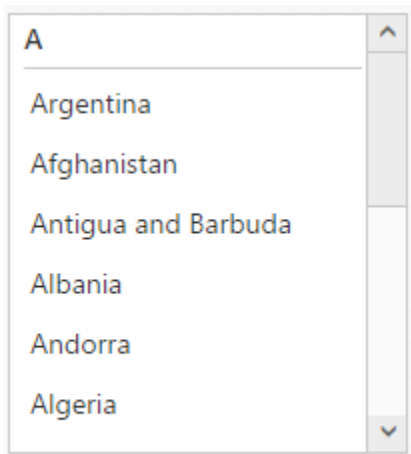
#### [Data Binding.](#)

The grouping will be defined based on the “groupBy” API in fields object.

### HTML

```
<ul id="listbox"></ul>
<script type="text/javascript">
$(function () {
//datasource for listbox
//Here the category column is used to define the grouping
var skillSet = [{ skill: "Bahrain", category: "B" },
{ skill: "Brazil", category: "B" },
{ skill: "Argentina", category: "A" },
{ skill: "Bangladesh", category: "B" },
{ skill: "Burma", category: "B" },
{ skill: "Afghanistan", category: "A" },
{ skill: "Antigua and Barbuda", category: "A" },
{ skill: "Barbados", category: "B" },
{ skill: "Botswana", category: "B" },
{ skill: "Albania", category: "A" },
{ skill: "Andorra", category: "A" },
{ skill: "Belarus", category: "B" },
{ skill: "Bolivia", category: "B" },
{ skill: "Algeria", category: "A" },
{ skill: "Angola", category: "A" }];
var listboxInstance = new ej.ListBox($("#listbox"), {
dataSource: skillSet,
fields: {
text: "skill",
//based on this field, grouping will be defined
groupBy: "category"
},
});
});
```

```
</script>
```



## Templates

The ListBox widget's appearance can be customized based on different needs using templates. The desired templates can be defined using the "template" property.

### HTML

```
<ul id="listbox"></ul>
<script type="text/javascript">
var data = [{
  text: "Erik Linden",
  imageName: "3",
  designation: "Representative",
  country: "England"
},
{ text: "John Linden", imageName: "6", designation: "Representative",
  country: "Norway" },
{ text: "Louis", imageName: "7", designation: "Representative", country:
  "Australia" },
{ text: "Lawrence", imageName: "8", designation: "Representative", country:
  "India" }];
$(function () {
  var listboxInstance = new ej.ListBox($("#listbox"), {
    dataSource: data,
    height: "240",
    width: "350",
    //defining templates
    template: '<div>' + '<div class="text"> ${text} </div><div
    class="desig">${designation}</div><div class="country"> ${country}
    </div></div>'
  });
});
</script>
```



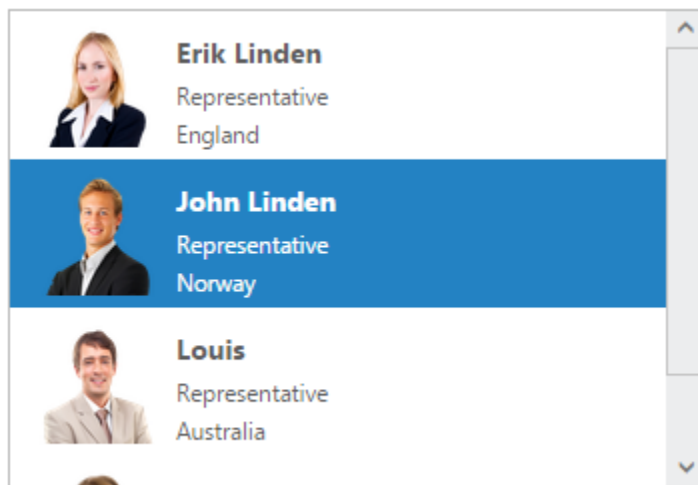
**Note:** In the above code snippet, the image path (images/Employees) is given just for demonstration. Hence the images will not be displayed while using the above code.

### Data Binding.

Define the styles for the template as below.

### CSS

```
.image
{
margin:0;
padding: 3px 10px 3px 3px;
border:0 none;
width:60px;
height:60px;
float:left;
}
.text
{
font-weight:bold;
padding:6px 3px 1px 3px;
}
.desig,.country{
font-size:smaller;
padding:3px 3px 0px 0px;
}
```



### Cascading

We can dynamically populate data of a list box while selecting an item in another list box i.e. rendering child list box based on the item selection in parent list box. This can be achieved using “cascadeTo” property.

Create the UL elements to render both the parent and the child ListBox widget as below.

### HTML

```
<style>
.parentlistbox, .childlistbox {
```

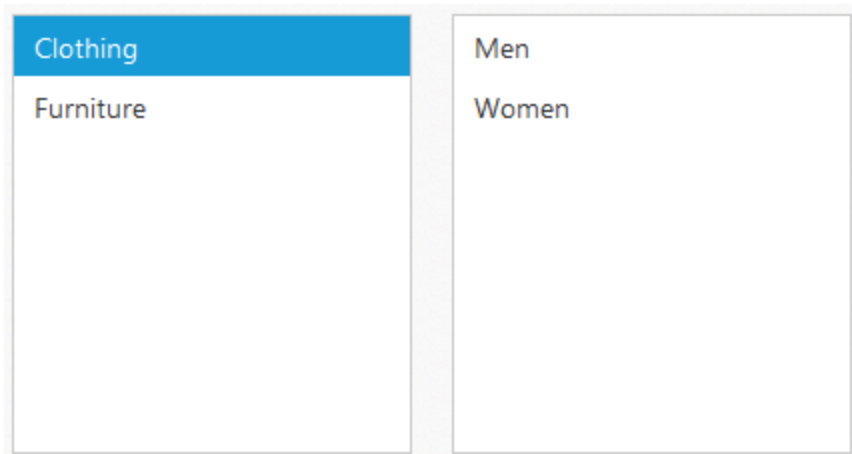
```
padding: 10px;
float: left;
}
</style>
<div class="controlitem">
<!--parent listbox element-->
<ul id="category"></ul>
</div>
<div class="controlitem">
<!-- child listbox element-->
<ul id="subcategoryList"></ul>
</div>
```

The parent child relationship should be defined in data sources of both the list boxes. I.e. both data sources should contain a common field for mapping (just like [primary key and foreign key definitions](#)).

Create the ListBox widgets as below.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ListBoxComponent {
$(function () {
// datasource for parent listbox
var data = [
{ categoryId: 'a', text: "Clothing" },
{ categoryId: 'b', text: "Furniture" }
];
//datasource for child listbox
var firstLevelChildData = [{ subCategoryId: 11, categoryId: 'a', text: "Men"
},
{ subCategoryId: 12, categoryId: 'a', text: "Women" },
{ subCategoryId: 13, categoryId: 'b', text: "Home furniture" },
{ subCategoryId: 14, categoryId: 'b', text: "Bedding" },
];
//create parent listbox
//cascadeTo property specifies the id of the child listbox (first level)
var listBoxInstance = new ej.ListBox($("#category"), {
dataSource: data,
fields: { value: "categoryId" },
cascadeTo: 'subcategoryList'
});
//create child listbox
//cascadeTo property specifies the id of the child listbox
var sublistboxInstance = new ej.ListBox($("#subcategoryList"), {
dataSource: firstLevelChildData,
loadDataOnInit: false
});
});
}
```



The parent ListBox widget's "cascadeTo" API should point to its child ListBox widget by specifying the id of the child ListBox widget. The child ListBox widget can be displayed with empty data on initialize by setting its "loadDataOnInit" property to false.

**Note:** In the below data source definition, the "categoryId" column will act as a primary key to define the parent-child relationship.

### Multilevel cascading

Please refer the below code snippets which is expanded from the above example, to achieve multi-level (three level here) cascading of the ListBox widgets.

Create the UL elements to render the parent and the child ListBox widgets as below.

### HTML

```
<style>
.controlitem {
float: left;
padding: 10px;
}
</style>
<div class="controlitem">
<!--parent listbox element-->
<ul id="category"></ul>
</div>
<div class="controlitem">
<!--first level child listbox element-->
<ul id="subcategoryList"></ul>
</div>
<div class="controlitem">
<!--second level child listbox element-->
<ul id="productList"></ul>
</div>
<div class="controlitem">
<!--third level child listbox element-->
<ul id="subproductList"></ul>
</div>
```

Create the ListBox widgets as below.

### JAVASCRIPT

```
$(function () {
    // datasource for parent listbox
    var data = [
        { categoryId: 'a', text: "Clothing" },
        { categoryId: 'b', text: "Furniture" }
    ];
    //datasource for first level child
    var firstLevelChildData = [{ subCategoryId: 11, categoryId: 'a', text: "Men"
    },
    { subCategoryId: 12, categoryId: 'a', text: "Women" },
    { subCategoryId: 13, categoryId: 'b', text: "Home furniture" },
    { subCategoryId: 14, categoryId: 'b', text: "Bedding" },
    ];
    //datasource for second level child
    var secondLevelChildData = [{ productid: 101, subCategoryId: 11, text: "men
    shirts" },
    { productid: 102, subCategoryId: 11, text: "men pants" },
    { productid: 103, subCategoryId: 12, text: "Women shirts" },
    { productid: 104, subCategoryId: 12, text: "Women pants" },
    { productid: 105, subCategoryId: 13, text: "sofa" },
    { productid: 106, subCategoryId: 13, text: "chairs" },
    { productid: 107, subCategoryId: 14, text: "bedsheets" },
    { productid: 108, subCategoryId: 14, text: "pillows" },
    ];
    //datasource for third level child
    var thirdLevelChildData = [{ productid: 101, text: "red men shirts" },
    { productid: 101, text: "blue men shirts" },
    { productid: 102, text: "red men pants" },
    { productid: 102, text: "blue men pants" },
    { productid: 103, text: "blueWomen shirts" },
    { productid: 103, text: "red Women shirts" },
    { productid: 104, text: "red women pants" },
    { productid: 104, text: "blue women pants" },
    { productid: 105, text: "red sofa" },
    { productid: 105, text: "blue sofa" },
    { productid: 106, text: "red chairs" },
    { productid: 106, text: "blue chairs" },
    { productid: 107, text: "red bedsheets" },
    { productid: 107, text: "blue bedsheets" },
    { productid: 108, text: "red pillows" },
    { productid: 108, text: "blue pillows" }
    ]
    //create parent listbox
    //cascadeTo property specifies the id of the child listbox (first level)
    var categoryInstance = new ej.ListBox($("#category"), {
        dataSource: data,
        fields: { value: "categoryId" },
        cascadeTo: 'subcategoryList'
    });
    //create first level child listbox
    //cascadeTo property specifies the id of the child listbox (second level)
    var subcategoryListInstance = new ej.ListBox($("#subcategoryList"), {
        dataSource: firstLevelChildData,
```

```
fields: { value: "subCategoryId" },
loadDataOnInit: false,
cascadeTo: 'productList'
});
//create second level child listbox
//cascadeTo property specifies the id of the child listbox (third level)
var productListInstance = new ej.ListBox($("#productList"), {
dataSource: secondLevelChildData,
fields: { value: "productid" },
loadDataOnInit: false,
cascadeTo: 'subproductList'
});
//create third level child listbox
var productListInstance = new ej.ListBox($("#subproductList"), {
dataSource: thirdLevelChildData,
loadDataOnInit: false,
});
});
```



## Drag and drop

A list item can be moved from a widget to another ListBox widget. Also the order of list items can be changed. This can be achieved using the drag and drop support.

### Transferring a ListBox data to another ListBox

In some scenarios we might want to transfer a ListBox data to another ListBox. In the below steps we will see how to move skills from a ListBox widget to another.

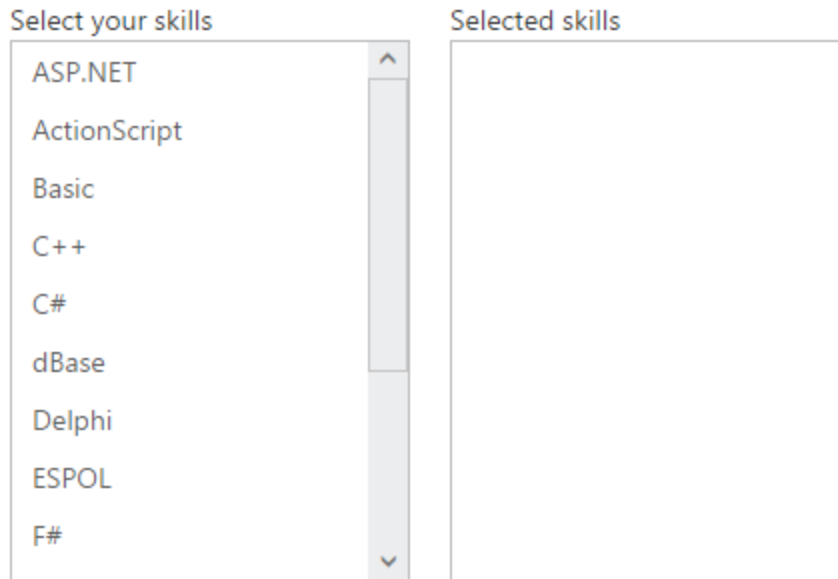
Enable the drag and drop support through “allowDrag” and “allowDrop” properties.

### HTML

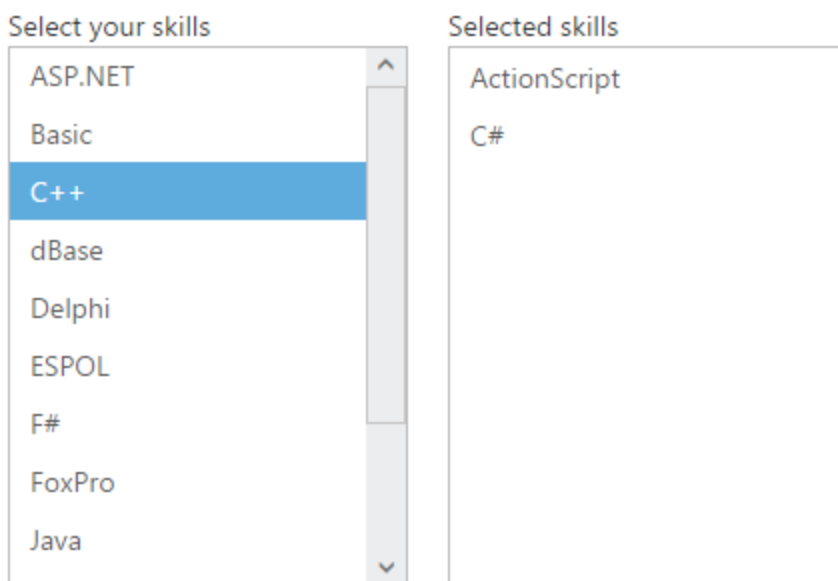
```
<!--listbox1-->
<div class="control">
<div>Select your skills</div>
<ul id="listbox1"></ul>
</div>
<!--listbox2-->
<div class="control">
<div>Selected skills</div>
<ul id="listbox2"></ul>
</div>
<script type="text/javascript">
/// <reference path="tsfiles/jquery.d.ts" />
```

```
/// <reference path="tsfiles/ej.web.all.d.ts" />\nmodule ListBoxComponent {\n  $(function () {\n    // datasource\n    var skillSet = [\n      { skill: "ASP.NET" },\n      { skill: "ActionScript" },\n      { skill: "Basic" },\n      { skill: "C++" },\n      { skill: "C#" },\n      { skill: "dBase" },\n      { skill: "Delphi" },\n      { skill: "ESPOL" },\n      { skill: "F#" },\n      { skill: "FoxPro" },\n      { skill: "Java" },\n      { skill: "J#" },\n      { skill: "Lisp" },\n      { skill: "Logo" },\n      { skill: "PHP" }\n    ];\n    //create listBox1 - draggable\n    var listBoxInstance = new ej.ListBox($("#listbox1"), {\n      dataSource: skillSet,\n      fields: { text: "skill" },\n      allowDrag: true\n    });\n    //create listBox1 - droppable\n    var listBoxInstance = new ej.ListBox($("#listbox2"), {\n      allowDrop: true\n    });\n  });\n}\n</script>\n<!--define the styles-->\n<style type="text/css" class="cssStyles">\n  .control {\n    margin-left: 20px;\n    float: left;\n  }\n</style>
```

**Note:** The datasource is not set for the second ListBox widget. In the above example we have restricted listBox1 as draggable element and the listBox2 as droppable element. In this case we can't drag an item from listBox2 to listBox1. If we want to achieve two way drag and drop, we need to enable both allowDrag and allowDrop properties in both ListBox widgets configuration.



*Before Drag and Drop*



*After some items were dragged and dropped*

#### [Dynamically set data source on drag and drop](#)

In local data binding, while moving the specific item from a ListBox to another, its data source will be updated automatically along with the DOM (which is explained here). But in case of remote binding, only the DOM will be updated. We need to update the data source manually since it's not possible to

update the remote data source. So in this case, we can use “itemDrop” event to update the datasource of the second ListBox widget based on the dropped items.

Both the ListBox widgets are bound to a remote data source.

### HTML

```
<div class="control">
<ul id="listbox1"></ul>
</div>
<div class="control">
<ul id="listbox2"></ul>
</div>
<script type="text/javascript">
$(function () {
    //create data manager
    var dataManager1 = ej.DataManager({
        //ASP.NET Web API
        url: "http://mvc.syncfusion.com/UGService/api/Orders",
        crossDomain: true
    });
    var listBoxInstance = new ej.ListBox($("#listbox1"), {
        dataSource: dataManager1,
        fields: { text: "CustomerID" },
        allowDrag: true
    });
    //create data manager
    var dataManager2 = ej.DataManager({
        //OData service
        url: "http://mvc.syncfusion.com/Services/Northwnd.svc/"
    });
    // create query
    var query = ej.Query()
        .from("Customers").take(10);
    //create listbox2 - droppable
    $('#listbox2').ejListBox({
        dataSource: dataManager2,
        fields: { text: "CustomerID" },
        query: query,
        allowDrop: true,
        itemDrop: "updateDataSource"
    });
});
function updateDataSource(args) {
    //update the listbox2's datasource
}
</script>
<!--define the styles-->
<style type="text/css" class="cssStyles">
.control {
margin-left: 20px;
float: left;
}
</style>
```



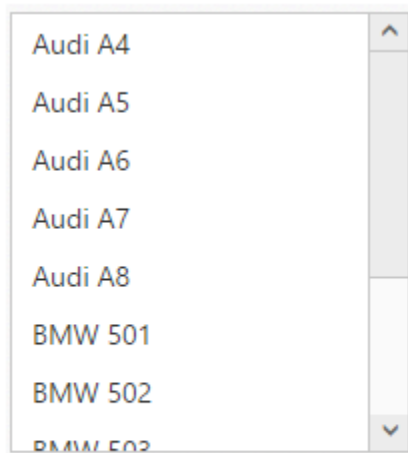
In the “itemDrop” event, we can implement “updateDataSource” function to update the datasource of the second ListBox widget. The “itemDrop” event’s argument contains the details of the dropped item.

### Reordering

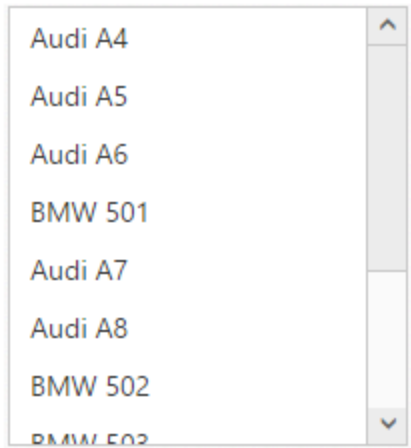
Item reordering can be done within a ListBox widget by enabling both “allowDrag” and “allowDrop” properties.

### HTML

```
<ul id="listbox">
<li>Audi A4</li>
<li>Audi A5</li>
<li>Audi A6</li>
<li>Audi A7</li>
<li>Audi A8</li>
<li>BMW 501</li>
<li>BMW 502</li>
<li>BMW 503</li>
<li>Batch</li>
<li>BMW 507</li>
<li>BMW 3200</li>
<li>Cut</li>
</ul>
<script type="text/javascript">
$(function () {
var listboxInstance = new ej.ListBox($("#listbox"), {
{
allowDrag: true,
allowDrop: true
});
});
});
</script>
```



*Before reordering*



*After reordering*

**Note:** The item reordering can be done dynamically without mouse interaction. For that we have provided two APIs namely “[moveUp](#)” and “[moveDown](#)”.

### Keyboard interaction

You can use Keyboard shortcut keys as an alternative for mouse actions to interact with the ListBox widget. Please refer the below table for details about short cut keys and its corresponding usage.

| Shortcut Key | Usage                    |
|--------------|--------------------------|
| Enter        | Selects the focused item |
| Up           | Moves to previous item   |
| Down         | Moves to next item       |
| Left         | Moves to previous item   |
| Right        | Moves to next item       |
| Home         | Moves to first item      |
| End          | Moves to last item       |

**Note:** Initial focus can be done by pressing tab key multiple times until it is focused.

### Incremental Search

The [Incremental search](#) helps in finding the specific item in the ListBox, as the user types the text one or more possible matches for the text are found and the first matched item will be selected. It can be enabled in the ListBox widget using “enableIncrementalSearch” API. The search can be case sensitive or case insensitive.

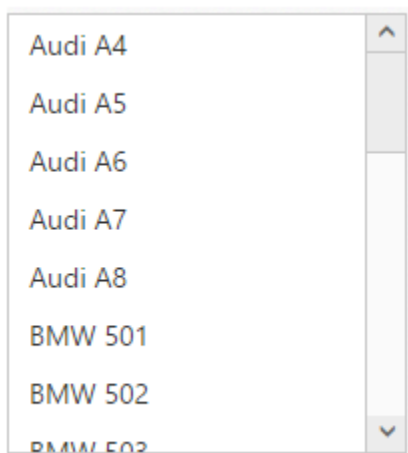
### HTML

```
<ul id="listbox">
  <li>Audi A4</li>
  <li>Audi A5</li>
```

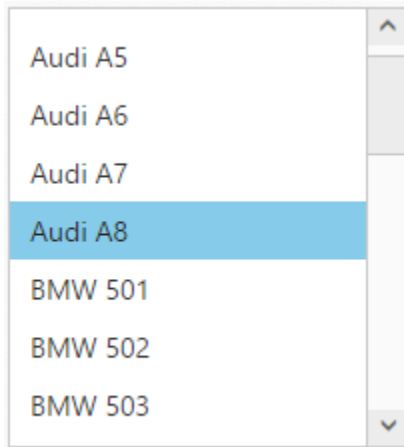
```

<li>Audi A6</li>
<li>Audi A7</li>
<li>Audi A8</li>
<li>BMW 501</li>
<li>BMW 502</li>
<li>BMW 503</li>
<li>Batch</li>
<li>BMW 507</li>
<li>BMW 3200</li>
<li>Cut</li>
<li>Copy</li>
<li>Paste</li>
<li>Add</li>
<li>Delete</li>
<li>D5ishCover</li>
<li>Di5shCover</li>
<li>Dis5hCover</li>
<li>Dish5Cover</li>
<li>DishC5over</li>
<li>DishCo5ver</li>
<li>DishCov5er</li>
<li>DishCove5r</li>
<li>Comments</li>
<li>Links</li>
<li>Clear Formatting</li>
</ul>
<script type="text/javascript">
  /// <reference path="tsfiles/jquery.d.ts" />
  /// <reference path="tsfiles/ej.web.all.d.ts" />
  module ListBoxComponent {
    $(function () {
      var listBoxInstance = new ej.ListBox($("#listbox"), {
        enableIncrementalSearch: true,
        caseSensitiveSearch: true
      });
    });
  }
</script>

```



Press **tab** key to get **ListBox** focus and press **"A"** (enable caps lock or press shift + **"A"** since case sensitive search is enabled) to get the below output.



## ListView

### Overview

The **Essential Typescript ListView** widget builds interactive **ListView** interface. This control allows you to select an item from a list-like interface and display a set of data items in different layouts or views. Lists are used for displaying data, data navigation, result lists, and data entry.

### Key Features

- **AJAX Load:** Loads AJAX content in the ListView content.
- **Data binding:** Supports Data binding with JSON data and remote data.
- **Template support:** The ListView supports template content.
- **Group List:** The ListView supports group item list.
- **Check List:** The ListView supports check list.

### Getting Started

Using the following steps, you can create a **Typescript** ListView component. The basic rendering of **Typescript** ListView is achieved with default functionality.

#### Create a ListView

You can create a **Typescript** application with the help of the given [Typescript Getting Started Documentation](#).

Within an index.html file and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Typescript Application</title>
```

```
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.web
.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add ListView here-->
</body>
</html>
```

Add a **<div>** element. It is a container for **ListView** control. To create the **ListView**, you should call the **ejListView** jQuery plug-in function.

### HTML

```
<div id="listview">
<ul>
<li data-ej-text="Inbox"></li>
<li data-ej-text="VIP"></li>
<li data-ej-text="Drafts"></li>
<li data-ej-text="Sent"></li>
<li data-ej-text="Junk"></li>
<li data-ej-text="All mails"></li>
<li data-ej-text="Mail"></li>
</ul>
</div>
<script src="app.js"></script>
```

Initialize the **ListView** in **app.ts** file by using the **ej.ListView** method.

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ListComponent {
$(function () {
var sample = new ej.ListView($("#listview"), {
});
}
```

Now build your application, so that the **app.js** file is automatically generated and got added to your project (User have nothing to do with this file). Now, whatever code changes that you make in **app.ts** file will be reflected in **app.js** file automatically.

Run the above code to render the following output:

|           |
|-----------|
| Inbox     |
| VIP       |
| Drafts    |
| Sent      |
| Junk      |
| All mails |
| Mail      |

### Add Header

You can add a header for **ListView**. Refer to the following script.

### JS

```
module ListComponent {  
  $(function () {  
    let sample = new ej.ListView($("#listview"), {showHeader: true, headerTitle:  
      "Mailbox"});  
  });  
}
```

Run the above code to render the following output:

| Mailbox   |
|-----------|
| Inbox     |
| VIP       |
| Drafts    |
| Sent      |
| Junk      |
| All mails |
| Mail      |

*Note: You can find the ListView properties from the [API reference](#) document*

## Grouped List

### First Level Group List

The **ListView** widget can make as grouped list by setting the “**data-ej-enableGroupList**” attribute as “**True**”. This groups the set of items listed under **ul**. You can identify the grouped items with the header title specified respectively.

Refer the following code example.

## HTML

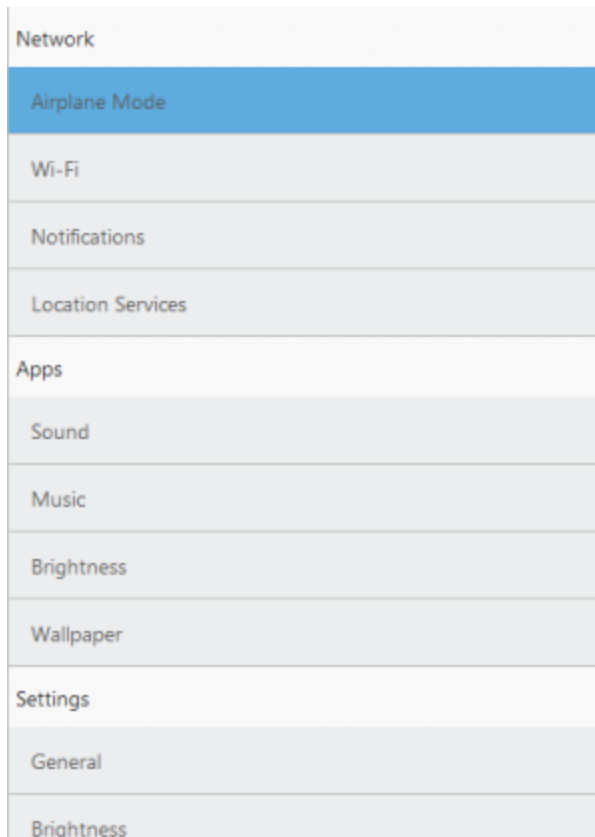
```
<div id="defaultListView" >
<ul data-ej-grouplisttitle="Network">
<li data-ej-text="Airplane Mode"></li>
<li data-ej-text="Wi-Fi"></li>
<li data-ej-text="Notifications"></li>
<li data-ej-text="Location Services"></li>
</ul>
<ul data-ej-grouplisttitle="Apps">
<li data-ej-text="Sound"></li>
<li data-ej-text="Music"></li>
<li data-ej-text="Brightness"></li>
<li data-ej-text="Wallpaper"></li>
</ul>
<ul data-ej-grouplisttitle="Settings">
<li data-ej-text="General"></li>
<li data-ej-text="Brightness"></li>
<li data-ej-text="Wallpaper"></li>
</ul>
</div>
```

Add the following script in your code.

## JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ListComponent {
$(function () {
var sample = new ej.ListView($("#defaultListView"), {
width:400,
enableGroupList:true
});
});
}
```

Run the codes to get the following output



### Nested Child Group List

While selecting a list item that is grouped, you can also render another set of list items. This is achieved by defining the desired **child item list** within the list containing **PrimaryKeyValue**. This **[data-ej-primarykey]** attribute relates the parent child for identifying its appropriate child when clicking on the parent list item.

Refer the following code examples.

### HTML

```
<div id="defaultListView" >
  <ul data-ej-grouplisttitle="Network">
    <li data-ej-text="Airplane Mode"></li>
    <li data-ej-text="Wi-Fi"></li>
    <li data-ej-text="Notifications"></li>
    <li data-ej-text="Location Services"></li>
  </ul>
  <ul data-ej-grouplisttitle="Apps">
    <li data-ej-primarykey="1" data-ej-text="Sound">
      <ul>
        <li data-ej-text="Ring Tone"></li>
        <li data-ej-text="Message Tone"></li>
        <li data-ej-text="Notification Tone"></li>
      </ul>
    </li>
    <li data-ej-text="Brightness"></li>
    <li data-ej-text="Wallpaper"></li>
  </ul>
</div>
```



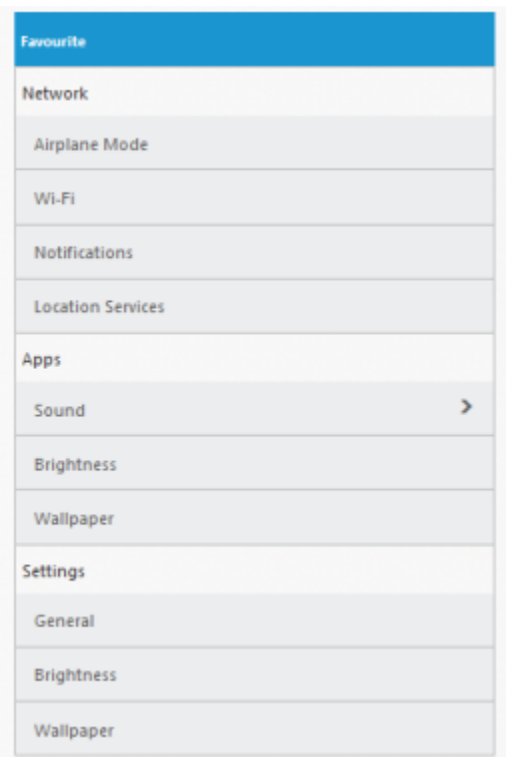
```
<ul data-ej-grouplisttitle="Settings">
<li data-ej-text="General"></li>
<li data-ej-text="Brightness"></li>
<li data-ej-text="Wallpaper"></li>
</ul>
</div>
```

Add the following script in your code.

### JAVASCRIPT

```
module ListComponent {
$(function () {
var sample = new ej.ListView($("#defaultListView"), {
showHeader: true,
headerTitle: "Favorite",
width:400,
enableGroupList:true
});
});
}
```

Run the codes to get the following output



### Selection

#### MultiSelection

**ListView** has a checklist feature that is used to select multiple list items at the same time in the **ListView**. For this, set **data-ej-enableCheckMark** attribute to **"True"**.

Refer the following code examples.

### HTML

```
<div id="defaultlistview">
<ul>
<li data-ej-text="Artwork"></li>
<li data-ej-text="Abstract"></li>
<li data-ej-text="2 Acrylic Mediums"></li>
<li data-ej-text="Creative Acrylic"></li>
<li data-ej-text="Modern Painting"></li>
<li data-ej-text="Canvas Art"></li>
<li data-ej-text="Black white"></li>
<li data-ej-text="Children"></li>
<li data-ej-text="Preschool Crafts"></li>
<li data-ej-text="School-age Crafts"></li>
</ul>
</div>
```

Add the following script in your code.

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ListComponent {
$(function () {
var sample = new ej.ListView($("#defaultlistview"), {
enableCheckMark: true,
width: 400
});
});
}
```

Run the codes to get the following output

|                   |                          |
|-------------------|--------------------------|
| Artwork           | <input type="checkbox"/> |
| Abstract          | <input type="checkbox"/> |
| 2 Acrylic Mediums | <input type="checkbox"/> |
| Creative Acrylic  | <input type="checkbox"/> |
| Modern Painting   | <input type="checkbox"/> |
| Canvas Art        | <input type="checkbox"/> |
| Black white       | <input type="checkbox"/> |
| Children          | <input type="checkbox"/> |
| Preschool Crafts  | <input type="checkbox"/> |
| School-age Crafts | <input type="checkbox"/> |

### PreventSelection

When selecting a specific list item, it is highlighted with an active color. **data-ej-preventSelection** attribute is used to prevent this behavior by setting it to **"True"**.

**Note:** When the click or select action is completed, the highlight is undone automatically even when the attribute is set to **"False"**.

Refer the following code examples.

#### HTML

```
<div id="defaultlistview">
<ul>
<li data-ej-text="Artwork"></li>
<li data-ej-text="Abstract"></li>
<li data-ej-text="2 Acrylic Mediums"></li>
<li data-ej-text="Creative Acrylic"></li>
<li data-ej-text="Modern Painting"></li>
<li data-ej-text="Canvas Art"></li>
<li data-ej-text="Black white"></li>
<li data-ej-text="Children"></li>
<li data-ej-text="Preschool Crafts"></li>
<li data-ej-text="School-age Crafts"></li>
</ul>
</div>
```

Add the following script in your code.

#### JAVASCRIPT

```
module ListComponent {
$(function () {
var sample = new ej.ListView($("#defaultlistview"), {
preventSelection: true,
width: 400
});
});
}
```

#### PersistSelection

**data-ej-persistSelection** attribute is used to highlight the selected item in the **ListView** control even after touch end happens. By default, the active state is removed once the touch end happens.

Refer the following code examples.

#### HTML

```
<div id="defaultlistview">
<ul>
<li data-ej-text="Artwork"></li>
<li data-ej-text="Abstract"></li>
<li data-ej-text="2 Acrylic Mediums"></li>
<li data-ej-text="Creative Acrylic"></li>
<li data-ej-text="Modern Painting"></li>
<li data-ej-text="Canvas Art"></li>
<li data-ej-text="Black white"></li>
<li data-ej-text="Children"></li>
```

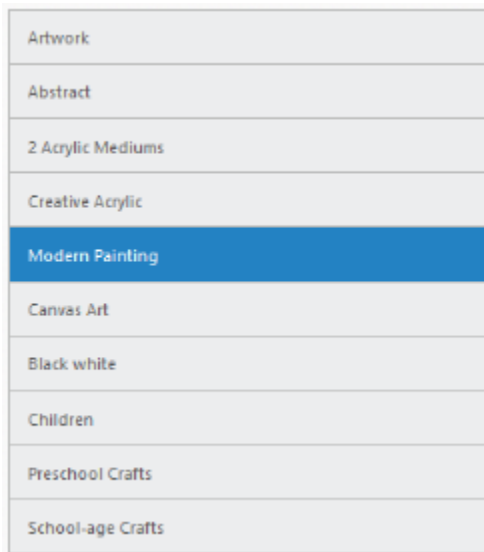
```
<li data-ej-text="Preschool Crafts"></li>
<li data-ej-text="School-age Crafts"></li>
</ul>
</div>
```

Add the following script in your code.

#### JAVASCRIPT

```
$(function () {
var sample = new ej.ListView($("#defaultlistview"), {
persistSelection: true,
width: 400
});
});
```

Run the codes to get the following output



#### Customize Header

In **ListView**, you can enable the built-in **Header** support. To show or hide the **Header** in **ListView**, use the `[data-ej-showheader]` attribute. By default, **ListView** is rendered with the **Header**. You can set the title for the **Header** by using the `[data-ej-headertitle]` attribute.

In some cases, for the purpose of navigation, you may want to show the **Back** button in **ListView Header**. To achieve this, `[data-ej-showheaderbackbutton]` attribute is used. By default, **ListView** is not rendered with the header back button in parent page. To customize the text shown in **ListView Header Back** button, the attribute `[data-ej-headerbackbuttontext]` is used.

Refer the following code example.

#### HTML

```
<div id="defaultListView">
<ul>
<li data-ej-text="Artwork"></li>
<li data-ej-text="Abstract"></li>
```

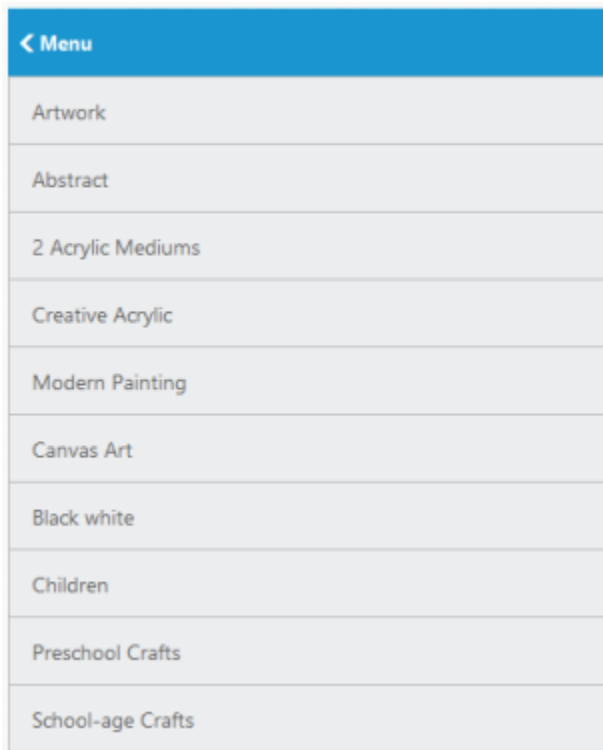
```
<li data-ej-text="2 Acrylic Mediums"></li>
<li data-ej-text="Creative Acrylic"></li>
<li data-ej-text="Modern Painting"></li>
<li data-ej-text="Canvas Art"></li>
<li data-ej-text="Black white"></li>
<li data-ej-text="Children"></li>
<li data-ej-text="Preschool Crafts"></li>
<li data-ej-text="School-age Crafts"></li>
</ul>
</div>
```

Add the following script in your code.

#### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ListComponent {
$(function () {
var sample = new ej.ListView($("#defaultListView"), {
showHeader:true,
showHeaderBackButton:true,
headerBackButtonText : "Menu",
width:400
});
});
}
```

Run the code to get the following output



## Data Binding

### Local Data Binding

**Essential Studio Web JS ListView** provides support for **Data Binding**. **Data Binding** provides a simple and consistent way for applications to present and interact with data. Elements can be bounded to data from a variety of data sources. In local data binding, the data source is written inside the program. Then it is handled by the **ListView** control. **DataSource** is used to get the **data source** that holds the list items.

Create div element to render the ListView sample.

### HTML

```
<div id="defaultlistview" ></div>
```

Add the following script in your code.

### JAVASCRIPT

```
window.dbitem =
[
{ "text": "Hot Singles" },
{ "text": "Rising Artists" },
{ "text": "Live Music" },
{ "text": "Best of 2013 So Far" },
{ "text": "100 Albums - $5 Each" },
{ "text": "Hip-Hop and R&B Sale" },
{ "text": "CD Deals" },
{ "text": "Songs" },
{ "text": "Bestselling Albums" },
{ "text": "New Releases" },
{ "text": "Bestselling Songs" },
{ "text": "Rock" },
{ "text": "Gospel" },
{ "text": "Latin Music" },
{ "text": "Jazz" },
{ "text": "Music Trade-In" },
{ "text": "Redeem a Gift Card" },
{ "text": "Band T-Shirts" },
{ "text": "Web MVC" }];
$(function () {
var sample = new ej.ListView($("#listview"), {
dataSource:window.dbitem,
width:400
});
});
```

Run the code to get the following output

|                       |
|-----------------------|
| Hot Singles           |
| Rising Artists        |
| Live Music            |
| Best of 2013 So Far   |
| 100 Albums - \$5 Each |
| Hip-Hop and R&B Sale  |
| CD Deals              |
| Songs                 |
| Bestselling Albums    |
| New Releases          |
| Bestselling Songs     |
| Rock                  |
| Gospel                |
| Latin Music           |
| Jazz                  |
| Music Trade-In        |
| Redeem a Gift Card    |
| Band T-Shirts         |

## Remote Data Binding

**Essential Studio Web JS ListView** provides support for **Data Binding**.

### *OData*

OData is a standardized protocol for creating and consuming the data. You can retrieve data from [OData service](#) by using **ej.DataManager**.

Here the **CustomerID** field is mapped with text property of the field object. The queries can be created using **ej.Query()**.

### **HTML**

```
<div id="listview"></div>
```

### **JAVASCRIPT**

```
var dataManger = ej.DataManager({ url:
"http://js.syncfusion.com/ejservices/Wcf/Northwind.svc/", crossDomain: true
});
var query = ej.Query().from('Customers').take(10);
$(function () {
var sample = new ej.ListView($("#listview"), {
datasource: dataManger,
fieldsettings: { "text": "CustomerID" },
query: query
});
});
```

Run the code to get the following output

|       |
|-------|
| ALFKI |
| ANATR |
| ANTON |
| AROUT |
| BERGS |
| BLAUS |
| BLONP |
| BOLID |
| BONAP |
| BOTTM |

#### *URL Adaptor*

URL Adaptor of **DataManager** can be used when you are required to use remote service to retrieve data. It interacts with server-side for all **DataManager** Queries and **CRUD** operations.

Now, in the following code example the data is retrieved from **MVC Controller**.

#### **HTML**

```
<div id="listview"></div>
```

#### **JAVASCRIPT**

```
var dataManager = ej.DataManager({ url: "Home/Data", adaptor: new
ej.UrlAdaptor() });
var query = ej.Query().take(10);
$(function () {
var sample = new ej.ListView($("#listview"), {
datasource: dataManager,
fieldsettings: { "text": "caption" },
query: query
});
});
```

#### **C#**

```
public partial class ListviewController : Controller
{
public JsonResult DataSource()
{
IEnumerable Data = setListSource();
return Json(Data, JsonRequestBehavior.AllowGet);
}
public static List<ListLocalData> setListSource()
{
```



```
List<ListLocalData> listSource = new List<ListLocalData>();  
listSource.Add(new ListLocalData { caption = "Hot Singles" });  
listSource.Add(new ListLocalData { caption = "Rising Artists" });  
listSource.Add(new ListLocalData { caption = "Live Music" });  
listSource.Add(new ListLocalData { caption = "Best of 2013 So Far" });  
listSource.Add(new ListLocalData { caption = "100 Albums - $5 Each" });  
listSource.Add(new ListLocalData { caption = "Hip-Hop and R&B Sale" });  
listSource.Add(new ListLocalData { caption = "CD Deals" });  
listSource.Add(new ListLocalData { caption = "Songs" });  
listSource.Add(new ListLocalData { caption = "Bestselling Albums" });  
listSource.Add(new ListLocalData { caption = "New Releases" });  
return listSource;  
}
```

Run the code to get the following output

|                       |
|-----------------------|
| Hot Singles           |
| Rising Artists        |
| Live Music            |
| Best of 2013 So Far   |
| 100 Albums - \$5 Each |
| Hip-Hop and R&B Sale  |
| CD Deals              |
| Songs                 |
| Bestselling Albums    |
| New Releases          |

### WebAPI

WebApi Adaptor, extended from ODataAdaptor, of DataManager is used for retrieving data from WebApi service.ASP.NET Web API is a Framework for building HTTP services. You can retrieve data from ASP.NET Web API by using **ej.DataManager**.

### HTML

```
<div id="listview"></div>
```

### JAVASCRIPT

```

<script>
var dataManger = ej.DataManager({
url: "http://js.syncfusion.com/ejservices/Wcf/Northwind.svc/Customers",
crossDomain: true,
adaptor: new ej.WebApiAdaptor()
});
var query = ej.Query();
$(function () {
var sample = new ej.ListView($("#listview"), {
datasource: dataManger,
fieldsettings: { "text": "CustomerID" },
query: query
});
});
</script>

```

Run the code to get the following output

|       |
|-------|
| ERNSH |
| FAMIA |
| FISSA |
| FOLIG |
| FOLKO |
| FRANK |
| FRANR |
| FRANS |
| FURIB |
| GALED |
| GODOS |

### FieldSettings

The **data-ej-FieldSettings** attribute is used to map the **DataSource** field with the list item fields. In addition to the list [item specific properties](#), the following fields are available while mapping.

#### FieldSettings

| Properties       | Definition  |
|------------------|---|
| ParentPrimaryKey | In DB, you can relate any child item to some other item. Set here is <code>~PrimaryKey</code> for the parent item. Here <code>~ParentPrimaryKey</code> defines the <code>~PrimaryKey</code> of some parent item to identify its parent. |
| Attributes       | In DB, you can define your desired class name or styles for the list item through the <code>~Attributes</code> field.   |

Please refer the following code examples.

**HTML**

```
<div id="defaultlistbox"></div>
```

Add the following script in your code.

**JAVASCRIPT**

```
window.dbitem =
[
  { "Texts": "Discover Music", "PrimaryKeys": "1", "Title": "Discover Music",
    "BackIconText": "back" },
  { "Texts": "Hot Singles", "ParentPrimaryKeyss": "1" },
  { "Texts": "Rising Artists", "PrimaryKeys": null, "ParentPrimaryKeyss": "1"
  },
  { "Texts": "Live Music", "ParentPrimaryKeyss": "1" },
  { "Texts": "Best of 2013 So Far", "ParentPrimaryKeyss": "1" },
  { "Texts": "Sales and Events", "PrimaryKeys": "2", "Title": "Sales and
Events", "BackIconText": "back" },
  { "Texts": "100 Albums - $5 Each", "ParentPrimaryKeyss": "2" },
  { "Texts": "Hip-Hop and R&B Sale", "ParentPrimaryKeyss": "2" },
  { "Texts": "CD Deals", "ParentPrimaryKeyss": "2" },
  { "Texts": "Categories", "PrimaryKeys": "3", "Title": "Categories",
    "BackIconText": "back" },
  { "Texts": "Songs", "ParentPrimaryKeyss": "3" },
  { "Texts": "Bestselling Albums", "ParentPrimaryKeyss": "3" },
  { "Texts": "New Releases", "ParentPrimaryKeyss": "3" },
  { "Texts": "Bestselling Songs", "ParentPrimaryKeyss": "3" },
  { "Texts": "MP3 Albums", "PrimaryKeys": "4", "Title": "MP3 Albums",
    "BackIconText": "back" },
  { "Texts": "Rock", "ParentPrimaryKeyss": "4" },
  { "Texts": "Gospel", "ParentPrimaryKeyss": "4" },
  { "Texts": "Latin Music", "ParentPrimaryKeyss": "4" },
  { "Texts": "Jazz", "ParentPrimaryKeyss": "4" },
  { "Texts": "More in Music", "PrimaryKeys": "5", "Title": "More in Music",
    "BackIconText": "back" },
  { "Texts": "Music Trade-In", "ParentPrimaryKeyss": "5" },
  { "Texts": "Redeem a Gift Card", "ParentPrimaryKeyss": "5" },
  { "Texts": "Band T-Shirts", "ParentPrimaryKeyss": "5" },
  { "Texts": "Web MVC", "ParentPrimaryKeyss": "5" }
];
window.musicFields = {
  "text": "Texts",
  "primaryKey": "PrimaryKeys",
  "parentPrimaryKey": "ParentPrimaryKeyss",
  "childHeaderTitle": "Title",
  "childHeaderBackButtonText": "BackIconText"
};
$(function () {
  var sample = new ej.ListView($("#defaultlistbox"), {
    fieldSettings: window.musicFields,
    dataSource: window.dbitem,
    width: 400,
    showHeader: true,
    headerTitle: "ListView"
  });
});
```

Run the code to get the following output



When you click on the parent item, it navigates to its corresponding child list item as follows.



## Filtering

**Filtering** is one of the key features of **ListView** control. The **Filtering** option is added into the **ListView** control when the **data-ej-enableFiltering** attribute is set to **"True"**. This enables a simple interface to filter items from a large collection of **ListView** items.

Refer the following code examples.

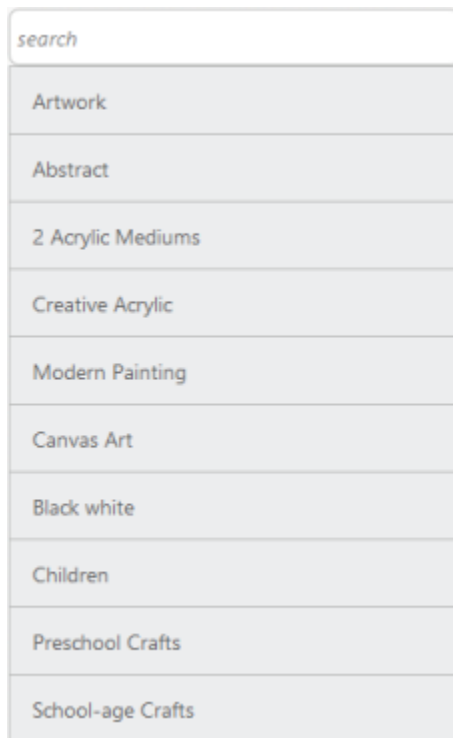
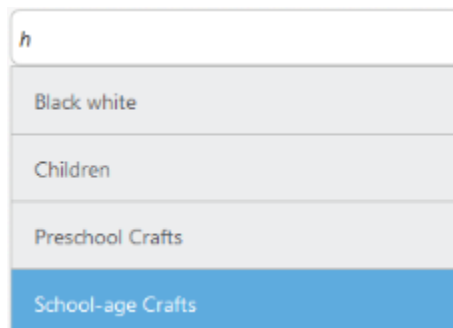
### HTML

```
<div id="defaultlistview">
<ul>
<li data-ej-text="Artwork"></li>
<li data-ej-text="Abstract"></li>
<li data-ej-text="2 Acrylic Mediums"></li>
<li data-ej-text="Creative Acrylic"></li>
<li data-ej-text="Modern Painting"></li>
<li data-ej-text="Canvas Art"></li>
<li data-ej-text="Black white"></li>
<li data-ej-text="Children"></li>
<li data-ej-text="Preschool Crafts"></li>
<li data-ej-text="School-age Crafts"></li>
</ul>
</div>
```

Add the following script in your code.

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ListComponent {
  $(function () {
    var sample = new ej.ListView($("#defaultlistview"), {
      width: 300,
      enableFiltering: true
    });
  });
}
```

**Screenshot:****Before Filtering****After Filtering****Dimensions**

To customize the **ListView** dimensions, **Width** and **Height** properties are used.

Refer to the following code examples.

### HTML

```
<div id="defaultlistview">
<ul>
<li data-ej-text="Artwork"></li>
<li data-ej-text="Abstract"></li>
<li data-ej-text="2 Acrylic Mediums"></li>
<li data-ej-text="Creative Acrylic"></li>
<li data-ej-text="Canvas Art"></li>
<li data-ej-text="Black white"></li>
<li data-ej-text="Children"></li>
<li data-ej-text="Preschool Crafts"></li>
<li data-ej-text="School-age Crafts"></li>
</ul>
</div>
```

Add the following script in your code.

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ListComponent {
$(function () {
var sample = new ej.ListView($("#defaultlistview"), {
width: 300,
height: 600
});
});
}
```

Run the code to get the following output

|                   |
|-------------------|
| Artwork           |
| Abstract          |
| 2 Acrylic Mediums |
| Creative Acrylic  |
| Canvas Art        |
| Black white       |
| Children          |
| Preschool Crafts  |
| School-age Crafts |

## Virtual Scrolling

We can load large data on demand using [allowVirtualScrolling](#) property. By default, "allowVirtualScrolling" set as boolean value of **"false"**. When it is set true, list items will be loaded on every scroll action. The number of items to be loaded per request can be specified using the [itemRequestCount](#) property. By default "itemRequestCount" value will be 5. We have provided two type of option for virtualScrolling,

### Normal Mode

This mode allows you to load the list view data while scrolling i.e. each time the scroll bar is scrolled, it will send request to the server to load the data.

### Continuous Mode

This mode allows you to load the list view data when the scrollbar reaches the end point. In this mode, we can specify the number of items to be loaded per request.

Refer the following code example.

#### HTML

```
<div class="cols-sample-area">
<div class="ctrllabel">Select a customer</div>
<div id="defaultlistbox">
</div>
</div>
```

Add the following script in your code.

#### JAVASCRIPT

```
var dataManger = ej.DataManager({
url: "http://js.syncfusion.com/ejservices/Wcf/Northwind.svc/", crossDomain:
true
});
// Query creation
var query = ej.Query()
.from("Customers");
$(function () {
var sample = new ej.ListView($("#defaultlistview"), {
dataSource: dataManger,
query: query,
fieldSettings: { text: "CustomerID" },
height:300,
itemRequestCount:8,
allowVirtualScrolling: true,
virtualScrollMode: ej.VirtualScrollMode.Continuous
});
});
```

## Maps

### Maps

A **Map** is a graphical representation of geographical data. It is used to represent the statistical data of a particular geographical area on Earth. Using pan and zoom, the maps can be navigated. Maps supports enhanced data visualization with bubbles and labels using data bound to the map.

## Use Case Scenarios

Some of the **Maps** control use case scenarios:

1. To visualize weather data.
2. To visualize geographical statistics information.
3. To visualize the density or availability of resources in an area.
4. To visualize political information.
5. To visualize the layout of a building.

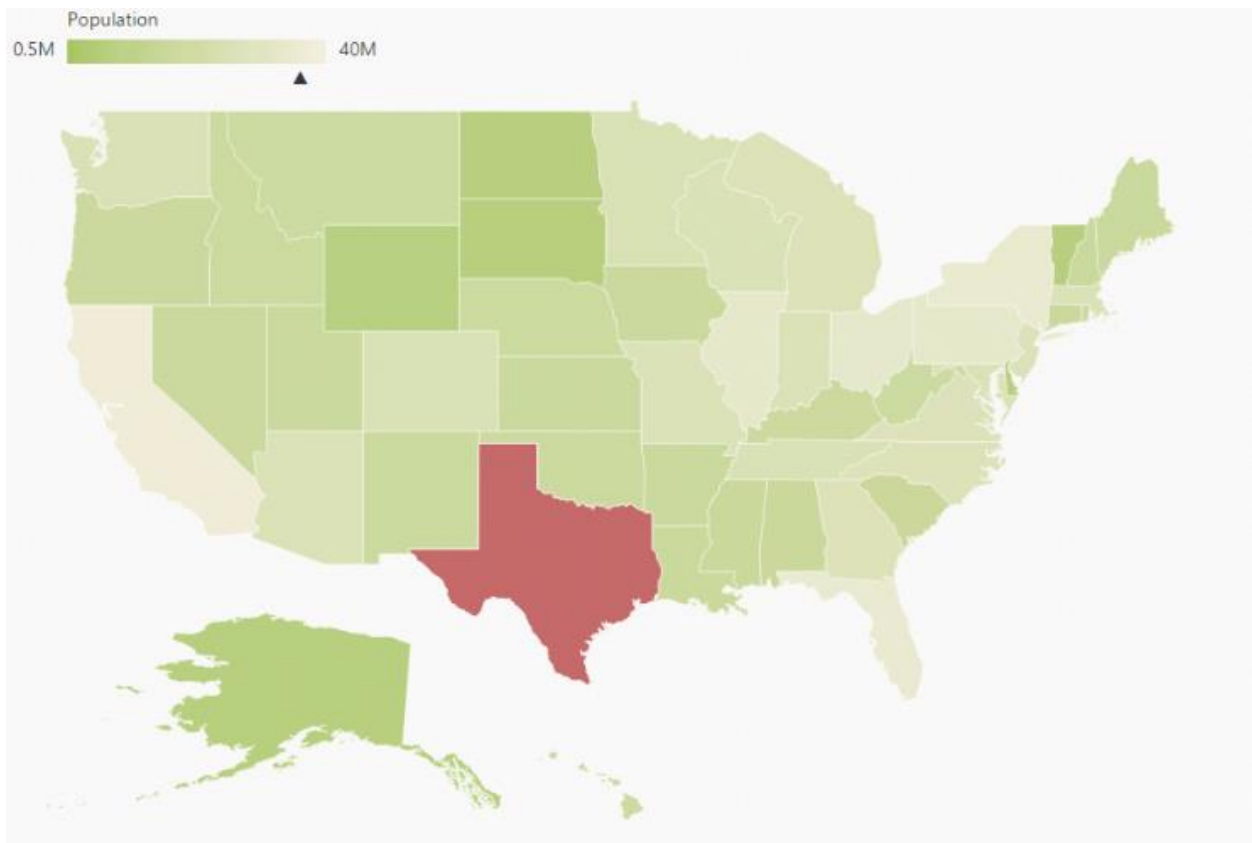
## Key Features

- **Layers** - Map is maintained through Layers and it can accommodate one or more layers.
- **Map Providers Support** - Syncfusion Map support map providers such as OpenStreetMap that can be added to any layers in maps.
- **GeoJSON Data Input** - The GeoJSON standard allows you to plot your own shapes using customized data to your maps. You can obtain some of the GeoJSON standard shape information from [GEOJSON format shapes](#) that can be used as data input for maps.
- **Map Elements Customization** - Map contains a set of elements, including shapes, bubbles, markers, legend, labels and data items that can be visualized with customized appearance showing additional information on the map using data bound data.
- **User Interaction** - Options like zooming, panning and map selection enables the effective interaction on map elements.

## Getting Started

- This section explains briefly about how to create Maps in your application with TypeScript
- You can learn how to configure Map with simple steps. In this example, you can learn how to configure USA population map with customized appearance and tooltip.





### Create a Map

You can easily create the Maps widget by using the following steps.

### Add Libraries

1.First create an TypeScript Project and add the following references in the app.ts file

For common getting started of TypeScript , you can refer [here](#).

The default type definition file **ej.web.all.d.ts** needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the intelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the **jquery.d.ts** file in your TypeScript application, which can be downloaded from [here](#).

2.Add the following script reference in the HTML page

### HTML

```
<!DOCTYPE html>
<html>
<head>
```

```
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="http://cdn.jsdelivr.net/jsrender/1.0pre35/jsrender.min.js"
type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

In the above code, `ej.web.all.min.js` script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

### Preparing Shape Data

The Shape Data collection describing geographical shape information can be obtained from [GEOJSON format shapes](#).

In this example, USA shape is used as shape data by utilizing the `United States of America.json` file in the following folder structure obtained from downloaded `Maps_GeoJSON` folder.

`..\ Maps_GeoJSON\All Countries with States`

You can assign the complete contents in “United States of America.json” file to new JSON object. For your better understanding, a JS file “usa.js” is already created to store JSON data in JSON object “usMap”.

### JAVASCRIPT

```
var usMap = //Paste all the content copied from the JSON file//
```

### Prepare Data Source

The datasource is populated with JSON data relative to shape data and stored in JSON object. USA population as datasource is used for better understanding.

The “populationData.js” file is used to store JSON data in JSON object “populationData”.

### JAVASCRIPT

```
var populationData = [
{ name: "California", population: "38332521" },
{ name: "Texas", population: "26448193" },
{ name: "New York", population: "19651127" },
{ name: "Florida", population: "19552860" },
{ name: "Illinois", population: "12882135" },
{ name: "Pennsylvania", population: "12773801" },
{ name: "Ohio", population: "11570808" },
{ name: "Georgia", population: "9992167" },
{ name: "Michigan", population: "9895622" },
{ name: "North Carolina", population: "9848060" },
{ name: "New Jersey", population: "8899339" },
```

```
[
  { name: "Virginia", population: "8260405" },
  { name: "Washington", population: "6971406" },
  { name: "Massachusetts", population: "6692824" },
  { name: "Arizona", population: "6626624" },
  { name: "Indiana", population: "6570902" },
  { name: "Tennessee", population: "6495978" },
  { name: "Missouri", population: "6044171" },
  { name: "Maryland", population: "5928814" },
  { name: "Wisconsin", population: "5742713" },
  { name: "Minnesota", population: "5420380" },
  { name: "Colorado", population: "5268367" },
  { name: "Alabama", population: "4833722" },
  { name: "South Carolina", population: "4774839" },
  { name: "Louisiana", population: "4625470" },
  { name: "Kentucky", population: "4395295" },
  { name: "Oregon", population: "3930065" },
  { name: "Oklahoma", population: "3850568" },
  { name: "Puerto Rico", population: "3615086" },
  { name: "Connecticut", population: "3596080" },
  { name: "Iowa", population: "3090416" },
  { name: "Mississippi", population: "2991207" },
  { name: "Arkansas", population: "2959373" },
  { name: "Utah", population: "2900872" },
  { name: "Kansas", population: "2893957" },
  { name: "Nevada", population: "2790136" },
  { name: "New Mexico", population: "2085287" },
  { name: "Nebraska", population: "1868516" },
  { name: "West Virginia", population: "1854304" },
  { name: "Idaho", population: "1612136" },
  { name: "Hawaii", population: "1404054" },
  { name: "Maine", population: "1328302" },
  { name: "New Hampshire", population: "1323459" },
  { name: "Rhode Island", population: "1051511" },
  { name: "Montana", population: "1015165" },
  { name: "Delaware", population: "925749" },
  { name: "South Dakota", population: "844877" },
  { name: "Alaska", population: "735132" },
  { name: "North Dakota", population: "723393" },
  { name: "District of Columbia", population: "646449" },
  { name: "Vermont", population: "626630" },
  { name: "Wyoming", population: "582658" }
]
```

You can refer to shape data and datasource as illustrated in the HTML page,

### HTML

```
<!-- Shape data file-->
<script src="usa.js" type="text/javascript"></script>
<!-- Data source file-->
<script src="populationData.js" type="text/javascript"></script>
```

### Initialize Map

1. Create a <div> tag and set the height and width to determine the map size.

## HTML

```
<div id="Map" style="width: 900px; height: 600px;"></div>
```

2. Initialize the Map in ts file by using the `ej.Map` method.

## JAVASCRIPT

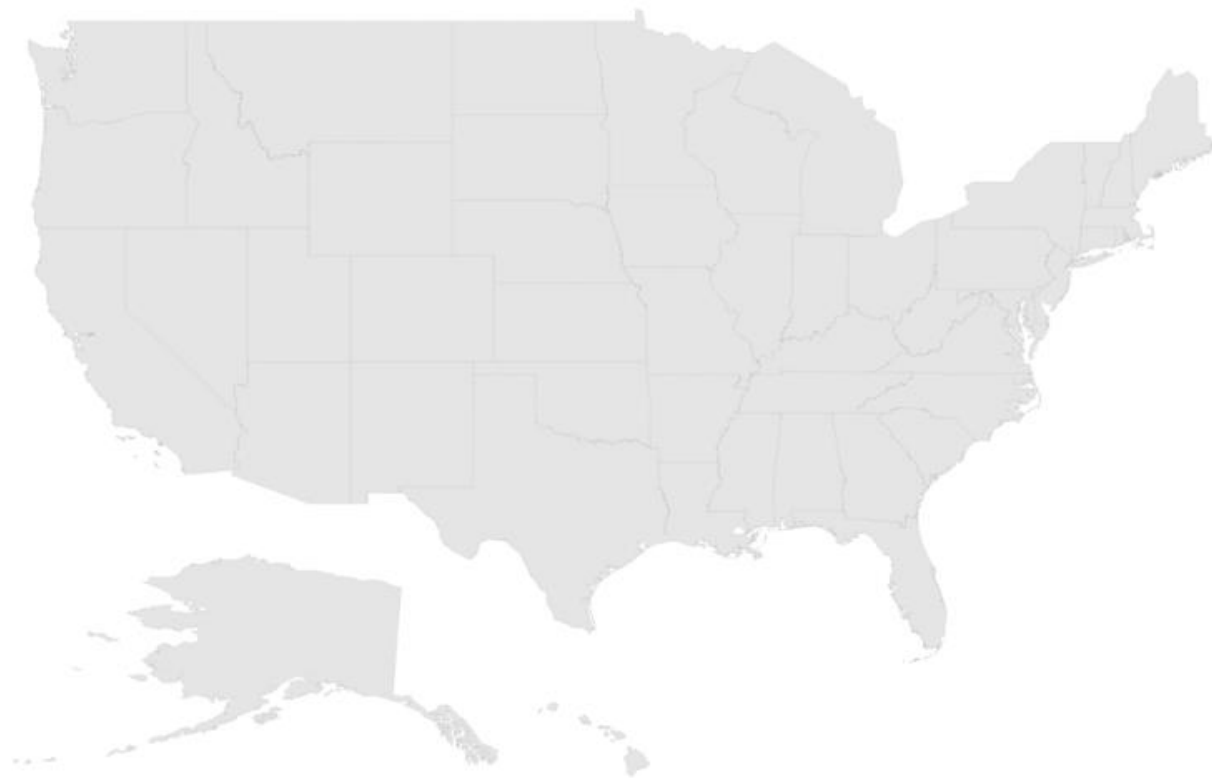
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MapComponent {
  $(function () {
    var MapSample = new ej.datavisualization.Map($("#Map"));
  });
}
```

3. Add the **shapeData** property in the maps to render it in layers

## JAVASCRIPT

```
module MapComponent {
  $(function () {
    var MapSample = new ej.datavisualization.Map($("#Map"), {
      layers: [
        {
          shapeData: usMap
        }
      ]
    });
  });
}
```

The above code renders a map with default properties and shape input provided through data in layers.



### Data Binding in Map

The following properties in shape layers is used for binding data in Maps control.

- DataSource
- ShapeDataPath
- ShapePropertyPath

#### DataSource

The `dataSource` property accepts collection values as input. For example, you can provide the list of objects as input.

#### Shape Data Path

The `shapeDataPath` property is used to refer the data ID in DataSource. For example, population MapData contains data ids 'Name' and 'Population'. The `shapeDataPath` and `shapePropertyPath` properties are related to each other (refer to `shapePropertyPath` for more details).

#### Shape Property Path

The `shapePropertyPath` property is similar to the `shapeDataPath` that refers the column name in the Data property of shape layers to identify the shape. When the values of the `shapeDataPath` property in the `dataSource` property and the value of `shapePropertyPath` in the Data property match, then the associated object from the `dataSource` is bound to the corresponding shape.

### JAVASCRIPT

```
$ (function () {
```

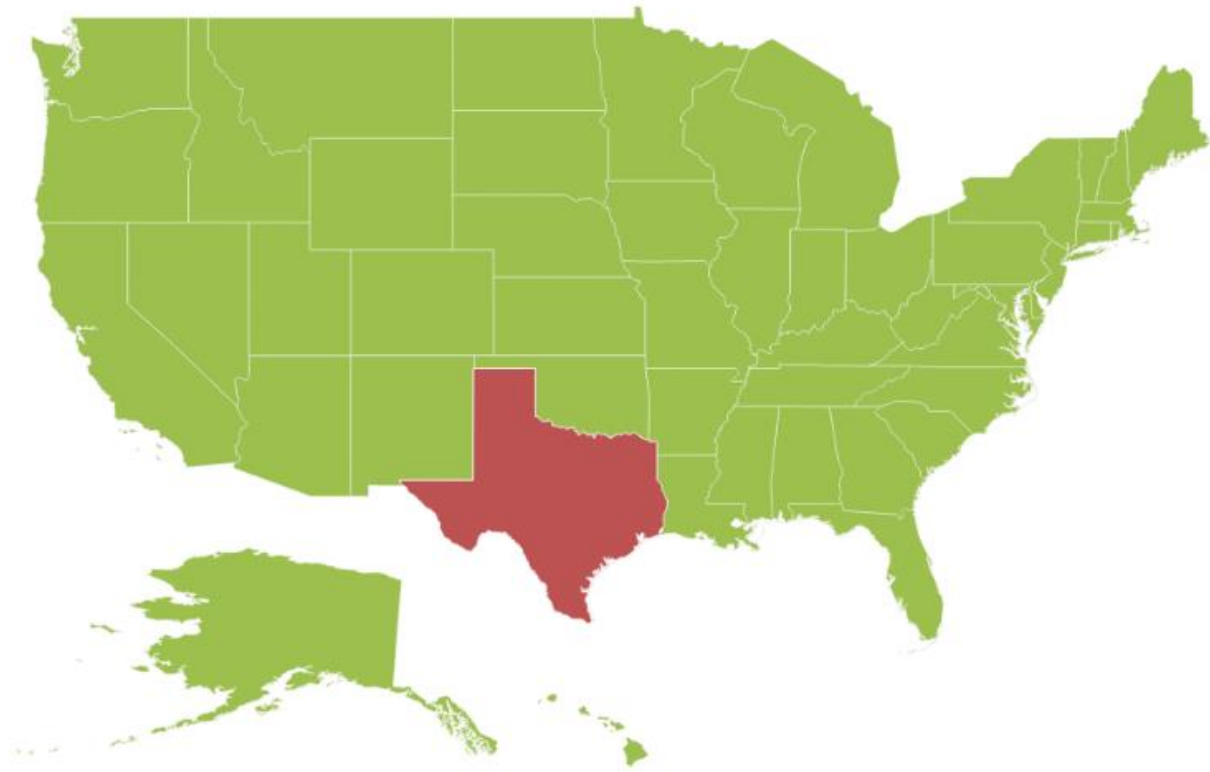
```
var MapSample = new ej.datavisualization.Map($("#Map"), {
  layers: [
    {
      shapeData: usMap,
      shapeDataPath: "name",
      shapePropertyPath: "name",
      dataSource: populationData
    }
  ]
});
```

### *Customize Map Appearance*

You can customize the shape's color by using `fill`, `stroke` and `strokeThickness` properties in `shapeSettings`.

### **JAVASCRIPT**

```
$(function () {
  var MapSample = new ej.datavisualization.Map($("#Map"), {
    layers: [
      {
        shapeData: usMap,
        shapeDataPath: "name",
        shapePropertyPath: "name",
        dataSource: populationData,
        enableSelection: false,
        enableMouseHover: true,
        shapeSettings: {
          fill: "#9CBF4E",
          strokeThickness: "0.5",
          stroke: "White",
          highlightStroke: "White",
          highlightColor: "#BC5353",
          highlightBorderWidth: "1"
        }
      }
    ]
  });
});
```



#### *Customizing Map Appearance by Range*

The Range color mapping is used to differentiate the shape's fill based on its underlying value and color ranges. The `from` and `to` properties defines the value ranges and the `gradientColors` property defines the equivalent color ranges respective to their value ranges.

---

**Note:** The `enableGradient` property value should be true to apply gradient colors for maps.

---

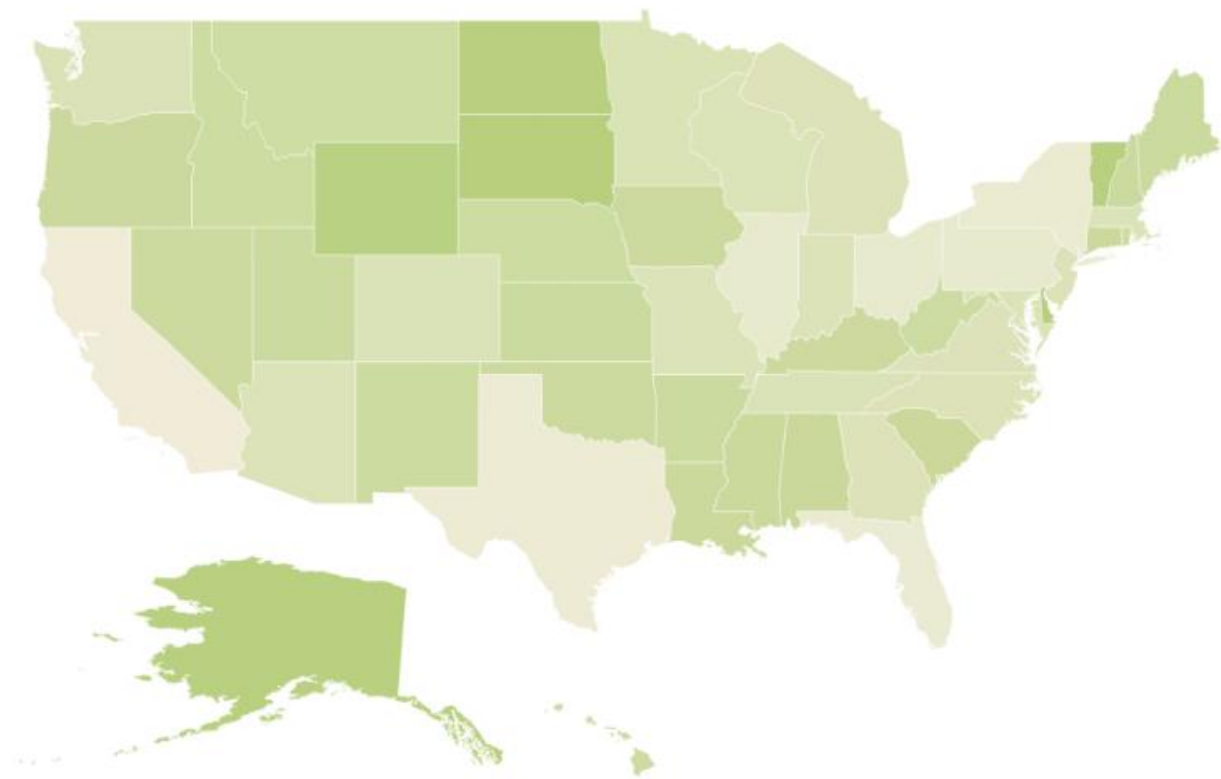
#### **JAVASCRIPT**

```
$(function () {  
  var MapSample = new ej.datavisualization.Map($("#Map"), {  
    layers: [  
      {  
        shapeData: usMap,  
        shapeDataPath: "name",  
        shapePropertyPath: "name",  
        dataSource: populationData,  
        enableSelection: false,  
        enableMouseHover: true,  
        shapeSettings: {  
          fill: "#9CBF4E",  
          strokeThickness: "0.5",  
          stroke: "White",  
          highlightStroke: "White",  
          highlightColor: "#BC5353",  
          highlightBorderWidth: "1",  
          valuePath: "name",  
          colorValuePath: "population",  
          enableGradient: true,  
        }  
      }  
    ]  
  });  
});
```

```
colorMappings:
{
  rangeColorMapping: [
    {
      from: 500000,
      to: 1000000,
      gradientColors: ["#9CBF4E", "#B8CE7B"]
    },
    {
      from: 1000001,
      to: 5000000,
      gradientColors: ["#B8CE7B", "#CBD89A"]
    },
    {
      from: 5000001,
      to: 10000000,
      gradientColors: ["#CBD89A", "#DEE2B9"]
    },
    {
      from: 10000001,
      to: 40000000,
      gradientColors: ["#DEE2B9", "#F1ECD8"]
    }
  ]
}
});
```

The following screenshot illustrates a Map with gradient color property enable.





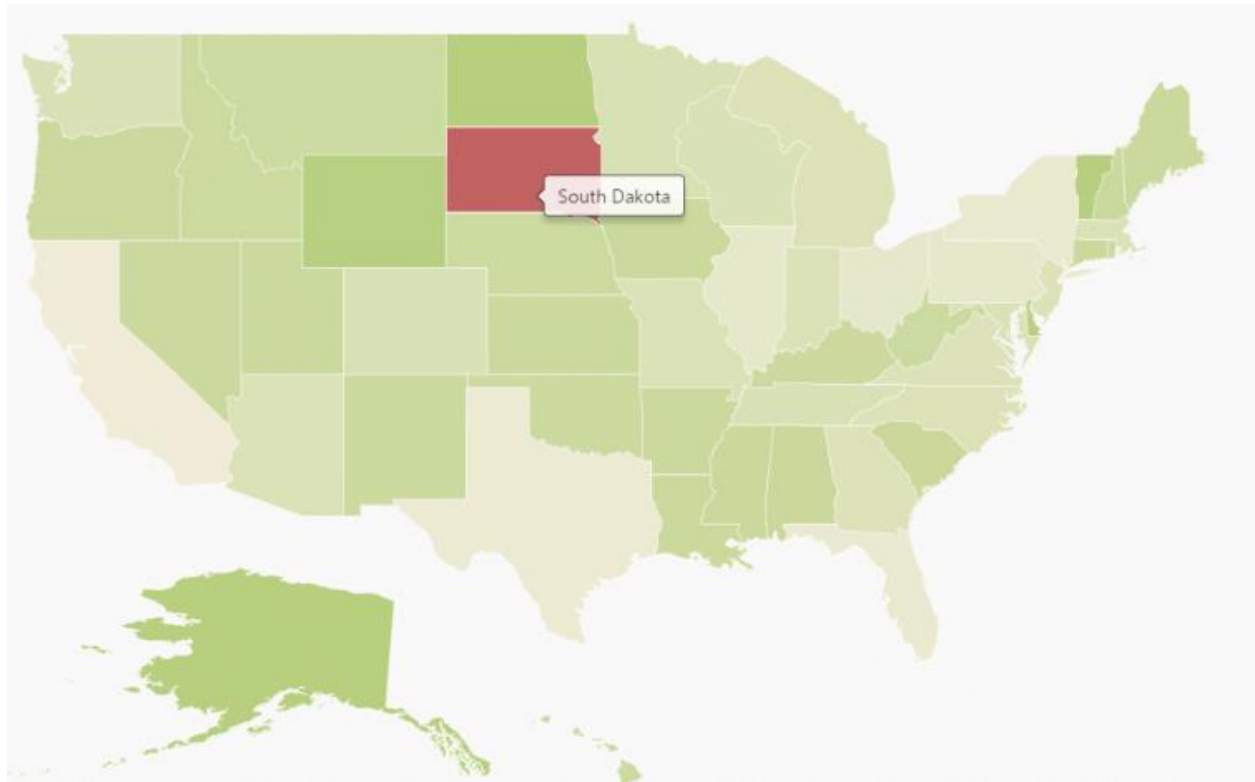
### Enable Tooltip

The tooltip is displayed only when `showTooltip` is set to 'true' in the shape layers. By default, it takes the property of the bound object that is referred in the `valuePath` and displays its content on hovering the corresponding shape. The `tooltipTemplate` property is used for customizing the template for tooltip.

### JAVASCRIPT

```
$(function () {
  var MapSample = new ej.datavisualization.Map($("#Map"), {
    layers: [
      {
        // ...
        shapeSettings: {
          // ...
          valuePath: "name",
          // ...
        },
        showTooltip: true
      }
    ]
  });
});
```

The following screenshot illustrates a map control displaying a Tooltip.



### Legend

A Legend can be made visible by setting the `showLegend` property in `legendSetting`.

#### Interactive Legend

The legends can be made interactive with an arrow mark indicating the exact range color in the legend, when the mouse hovers the corresponding shapes. You can enable this option by setting `mode` property in `legendSettings` value as 'Interactive'. The default value of `mode` property is 'Default' to enable the normal legend.

#### Title

Use `title` property to provide title for interactive legend.

#### Label

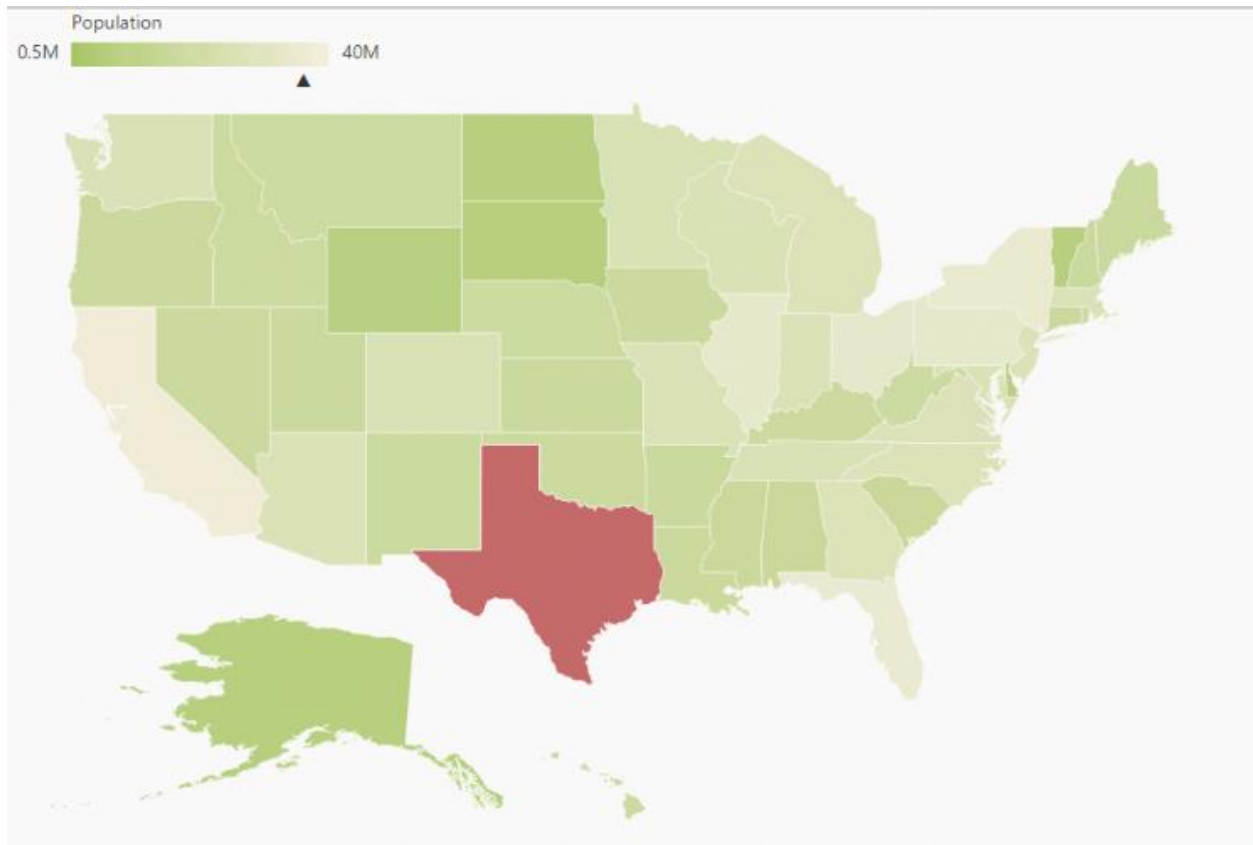
You can use `leftLabel` and `rightLabel` property to provide left and right labels for interactive legend.

### JAVASCRIPT

```
$(function () {  
  var MapSample = new ej.datavisualization.Map($("#Map"), {  
    layers: [  
      {  
        shapeData: usMap,  
        shapeDataPath: "name",  
        shapePropertyPath: "name",  
        dataSource: populationData,  
        enableSelection: false,  
        enableMouseHover: true,  
        shapeSettings: {  
          fill: "#9CBF4E",  
          strokeThickness: "0.5",
```

```
stroke: "White",
highlightStroke: "White",
highlightColor: "#BC5353",
highlightBorderWidth: "1",
valuePath: "population",
enableGradient: true,
colorMappings:
{
  rangeColorMapping: [
    {
      from: 500000,
      to: 1000000,
      gradientColors: ["#9CBF4E", "#B8CE7B"]
    },
    {
      from: 1000001,
      to: 5000000,
      gradientColors: ["#B8CE7B", "#CBD89A"]
    },
    {
      from: 5000001,
      to: 10000000,
      gradientColors: ["#CBD89A", "#DEE2B9"]
    },
    {
      from: 10000001,
      to: 40000000,
      gradientColors: ["#DEE2B9", "#F1ECD8"]
    }
  ]
},
legendSettings: {
  showLegend: true,
  dockOnMap: true,
  height: 18,
  width: 190,
  mode: "interactive",
  title: "Population",
  leftLabel: "0.5M",
  rightLabel: "40M"
}
});
});
```

The following screenshot illustrates a map displaying an interactive legend.



## Populate Data

In this section you can learn how to populate shape data for providing Data input to Map control and usage of DataSource property.

### Shape Data

The Shape Data collection describing geographical shape information can be obtained from [GEOJSON format shapes](#).

In this example, USA shape is used as shape data by utilizing the “**United States of America.json**” file in the following folder structure obtained from downloaded Maps\_GeoJSON folder.

```
..\Maps_GeoJSON\All Countries with States
```

You can assign the complete contents in “**United States of America.json**” file to new JSON object. For better understanding, a JS file “**usa.js**” is already created to store JSON data in JSON object “usMap”.

[usa.js]

### JAVASCRIPT

```
var usMap = //Paste all the content copied from the JSON file//
```

### Data Binding

The **Maps** control supports data binding with the `dataSource` property in the shape layers.

### Properties

The following properties in shape layers is be used for binding data in **Maps** control,

- dataSource
- shapeDataPath
- shapePropertyPath

#### Data Source

The `dataSource` property accepts the collection values as input. For example, you can provide the list of objects as input.

#### Shape Data Path

The `shapeDataPath` property is used to refer the data ID in DataSource. For example, population MapData contains data ids 'Name' and 'Population'. The `shapeDataPath` and the `shapePropertyPath` properties are related to each other (refer to `shapePropertyPath` for more details).

#### Shape Property Path

The `shapePropertyPath` property is similar to the `shapeDataPath` that refers to the column name in the `data` property of shape layers to identify the shape. When the values of the `shapeDataPath` property in the `dataSource` property and the value of `shapePropertyPath` in the `data` property match, then the associated object from the `dataSource` is bound to the corresponding shape.

The datasource is populated with JSON data relative to shape data and stored in JSON object. The USA population as datasource is used for better understanding.

The "populationData.js" file is used to store JSON data in JSON object "populationData".

Refer both shape data and datasource as illustrated in the following code example.

#### [populationData.js]

#### JAVASCRIPT

```
/// <reference path="../../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../../tsfiles/ej.web.all.d.ts"></reference>
var populationData = [
  { name: "California", population: "38332521" },
  { name: "Texas", population: "26448193" },
  { name: "New York", population: "19651127" },
  { name: "Florida", population: "19552860" },
  { name: "Illinois", population: "12882135" },
  { name: "Pennsylvania", population: "12773801" },
  { name: "Ohio", population: "11570808" },
  { name: "Georgia", population: "9992167" },
  { name: "Michigan", population: "9895622" },
  { name: "North Carolina", population: "9848060" },
  { name: "New Jersey", population: "8899339" },
  { name: "Virginia", population: "8260405" },
  { name: "Washington", population: "6971406" },
  { name: "Massachusetts", population: "6692824" },
  { name: "Arizona", population: "6626624" },
  { name: "Indiana", population: "6570902" },
  { name: "Tennessee", population: "6495978" },
  { name: "Missouri", population: "6044171" },
  { name: "Maryland", population: "5928814" },
  { name: "Wisconsin", population: "5742713" },
  { name: "Minnesota", population: "5420380" },
  { name: "Colorado", population: "5268367" },
]
```

```

{ name: "Alabama", population: "4833722" },
{ name: "South Carolina", population: "4774839" },
{ name: "Louisiana", population: "4625470" },
{ name: "Kentucky", population: "4395295" },
{ name: "Oregon", population: "3930065" },
{ name: "Oklahoma", population: "3850568" },
{ name: "Puerto Rico", population: "3615086" },
{ name: "Connecticut", population: "3596080" },
{ name: "Iowa", population: "3090416" },
{ name: "Mississippi", population: "2991207" },
{ name: "Arkansas", population: "2959373" },
{ name: "Utah", population: "2900872" },
{ name: "Kansas", population: "2893957" },
{ name: "Nevada", population: "2790136" },
{ name: "New Mexico", population: "2085287" },
{ name: "Nebraska", population: "1868516" },
{ name: "West Virginia", population: "1854304" },
{ name: "Idaho", population: "1612136" },
{ name: "Hawaii", population: "1404054" },
{ name: "Maine", population: "1328302" },
{ name: "New Hampshire", population: "1323459" },
{ name: "Rhode Island", population: "1051511" },
{ name: "Montana", population: "1015165" },
{ name: "Delaware", population: "925749" },
{ name: "South Dakota", population: "844877" },
{ name: "Alaska", population: "735132" },
{ name: "North Dakota", population: "723393" },
{ name: "District of Columbia", population: "646449" },
{ name: "Vermont", population: "626630" },
{ name: "Wyoming", population: "582658" }
]
<!-- Shape data file-->
<script src="usa.js" type="text/javascript"></script>
<!-- Data source file-->
<script src="populationData.js" type="text/javascript"></script>

```

The JSON object “populationData” is used as `dataSource` in the following code example.

### JAVASCRIPT

```

module MapComponent {
  $(function) ($) {
    var mapSample = new ej.datavisualization.Map($("#map"), {
      layers: [
        {
          shapeData: usMap,
          shapeDataPath: "name",
          shapeIdTableField: "name",
          dataSource: populationData
        }
      ]
    });
  }
}

```

## Customization

**Maps** control supports color customization to determine the exact combination of colors for shapes displayed in Maps and tooltip support to display additional information of shape data.

### Shape Settings

The `shapeSettings` defines the basic customization settings of shapes in the map.

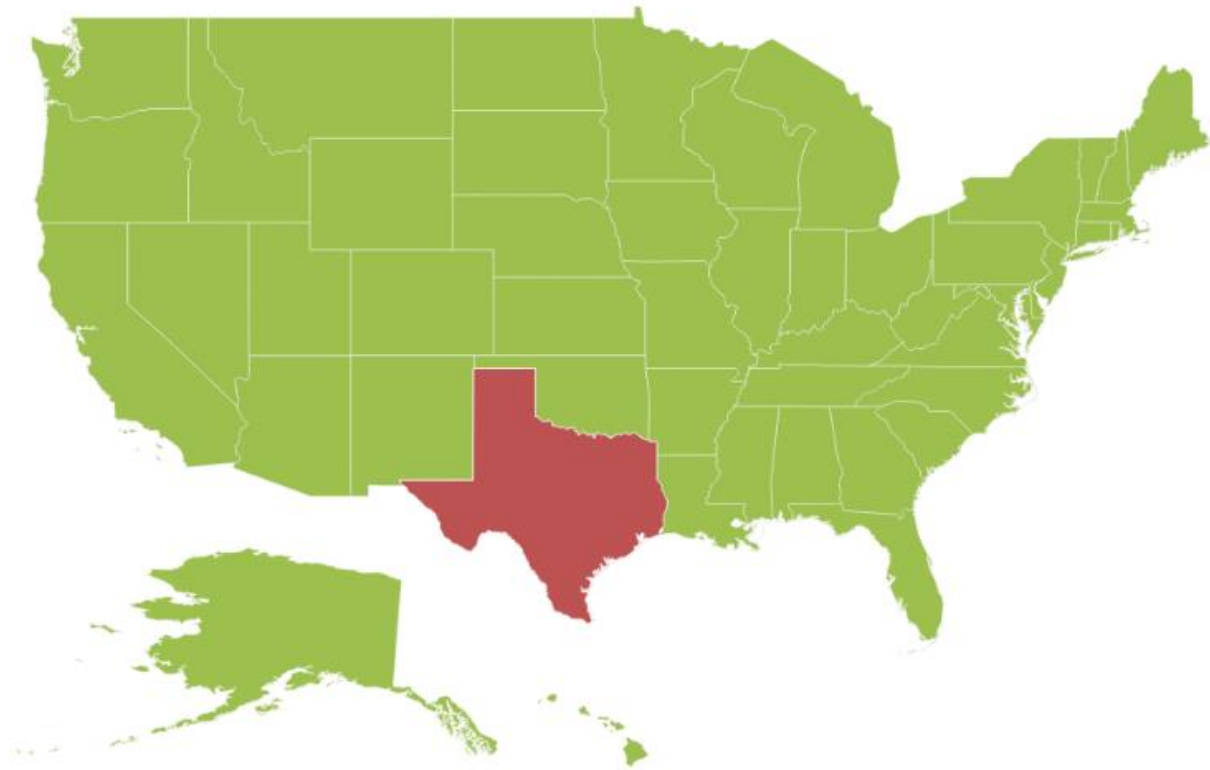
The important property that makes an impact on shape colors is the `autoFill`. This `autoFill` property is available in the `shapeSettings`.

- `fill` - It is used to set the fill color of the shapes in the map.
- `stroke` - It is used to set the border color of the shape in the map.
- `strokeThickness` - It is used to set the border thickness of the shape in the map.
- `highlightColor` - It is used to set the mouse hover color for shapes in the map.
- `highlightBorderWidth` - It is used to set the mouse hover border width for shapes in the map.
- `selectionColor` - It is used to set the selection color for shapes in the map.

The above properties of `shapeSettings` are applied only when the `autoFill` property value is set to false. By default, the `autoFill` property value is false.

### JAVASCRIPT

```
/// <reference path="../../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../../tsfiles/ej.web.all.d.ts"></reference>
module MapComponent {
  $(function () {
    var mapSample = new ej.datavisualization.Map($("#map"), {
      layers: [
        {
          shapeData: usMap,
          enableMouseHover: true,
          shapeSettings: {
            fill: "#9CBF4E",
            strokeThickness: "0.5",
            stroke: "White",
            highlightStroke: "White",
            highlightColor: "#BC5353",
            highlightBorderWidth: "1"
          }
        }
      ]
    });
  });
}
```



### Color Mapping

The **Color Mapping** support enables the customization of shape colors based on the underlying value of shape received from bounded data.

- **colorValuePath** - It renders the field value that is to be fetched from data for each shape used for determining the shape color.
- **valuePath**- It renders the field value that is to be fetched from data for each shape. This support also provides a tree map-like impact on the map UI. The various types of Color Mapping supported in maps are listed as follows.
- **rangeColorMapping** - It is used to differentiate the shape's fill based on its underlying value and color ranges. The properties of rangeColorMapping are listed in the following table.

| Property | Type   | Description  |
|----------|--------|--|
| from     | Double | Gets or sets start value   |
| to       | Double | Gets or sets end value   |
| color    | Color  | Gets or sets the colors to be applied for specific range value containing shapes when <b>enableGradient</b> property value is false. |
| label    | String | Gets or sets the label for legend when mode property value is "default".   |



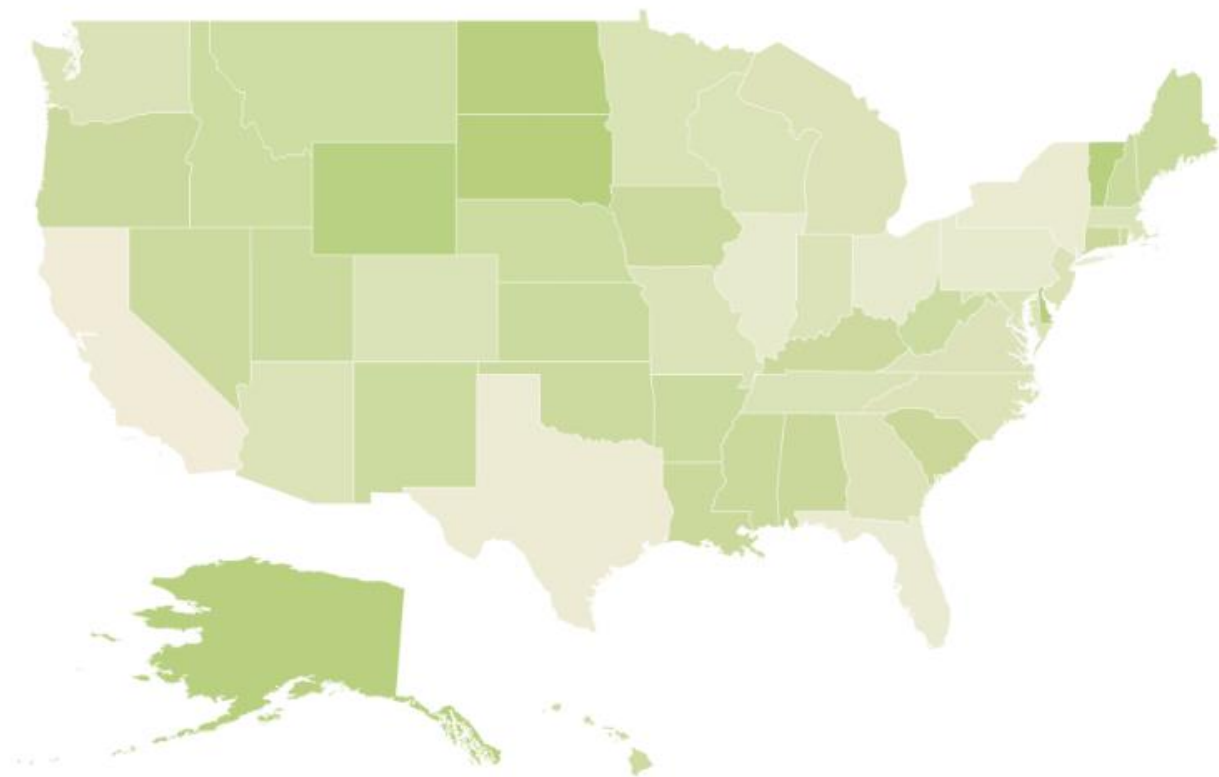
|                |       |   |
|----------------|-------|---|
| gradientColors | Array | Gets or sets the start point and end point gradient colors to be applied for specific range value containing shapes when <code>enableGradient</code> property value is set to true. |
|----------------|-------|---|

**JAVASCRIPT**

```
$(function () {
var mapSample = new ej.datavisualization.Map($("#map"), {
layers: [
{
shapeData: usMap,
shapeDataPath: "name",
shapePropertyPath: "name",
dataSource: populationData,
shapeSettings: {
fill: "#9CBF4E",
strokeThickness: "0.5",
stroke: "White",
valuePath: 'population',
enableGradient: true,
colorMappings:
{
rangeColorMapping: [
{
from: 500000,
to: 1000000,
gradientColors: ["#9CBF4E", "#B8CE7B"]
},
{
from: 1000001,
to: 5000000,
gradientColors: ["#B8CE7B", "#CBD89A"]
},
{
from: 5000001,
to: 10000000,
gradientColors: ["#CBD89A", "#DEE2B9"]
},
{
from: 10000001,
to: 40000000,
gradientColors: ["#DEE2B9", "#F1ECD8"]
}
]
}
}
}
]
});
});
```

When the underlying object value is 700000, then the fill color of the corresponding shape is set between #9CBF4E and #B8CE7B.

When the underlying value is below any of the given sorted range or above the sorted range, then the fill is set from fill.



- **equalColorMapping** - The equalColorMapping is used to differentiate the shape's fill based on its underlying value and color. The properties of equalColorMapping is listed in the following table.

| Property | Type   | Description                         |
|----------|--------|-------------------------------------|
| value    | String | Gets or sets the value.             |
| color    | String | Gets or sets the color for mapping. |

In equal color mapping, value property contains the values of the field set in **colorValuePath** property of shape settings.

Here USA election data is considered as input datasource and stored in "electionData.js".

#### JAVASCRIPT

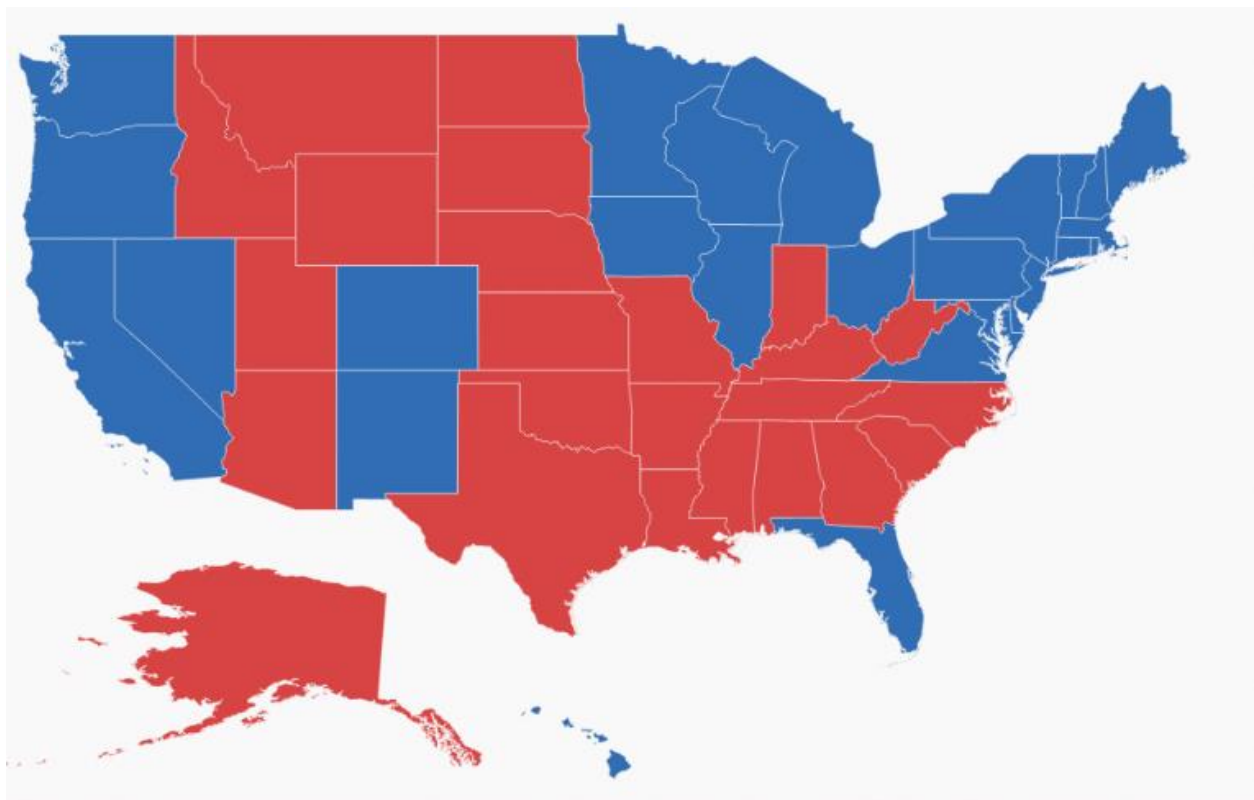
```
var electionData = [
  { State: "Alabama", Candidate: "Romney", Electors: 9 },
  { State: "Alaska", Candidate: "Romney", Electors: 3 },
  { State: "Arizona", Candidate: "Romney", Electors: 11 },
  { State: "Arkansas", Candidate: "Romney", Electors: 6 },
  { State: "California", Candidate: "Trump", Electors: 55 },
  { State: "Colorado", Candidate: "Trump", Electors: 9 },
  { State: "Connecticut", Candidate: "Trump", Electors: 7 },
  { State: "Delaware", Candidate: "Trump", Electors: 3 },
```

```

{ State: "District of Columbia", Candidate: "Trump", Electors: 3 },
{ State: "Florida", Candidate: "Trump", Electors: 29 },
{ State: "Georgia", Candidate: "Romney", Electors: 16 },
{ State: "Hawaii", Candidate: "Trump", Electors: 4 },
{ State: "Idaho", Candidate: "Romney", Electors: 4 },
{ State: "Illinois", Candidate: "Trump", Electors: 20 },
{ State: "Indiana", Candidate: "Romney", Electors: 11 },
{ State: "Iowa", Candidate: "Trump", Electors: 6 },
{ State: "Kansas", Candidate: "Romney", Electors: 6 },
{ State: "Kentucky", Candidate: "Romney", Electors: 8 },
{ State: "Louisiana", Candidate: "Romney", Electors: 8 },
{ State: "Maine", Candidate: "Trump", Electors: 4 },
{ State: "Maryland", Candidate: "Trump", Electors: 10 },
{ State: "Massachusetts", Candidate: "Trump", Electors: 11 },
{ State: "Michigan", Candidate: "Trump", Electors: 16 },
{ State: "Minnesota", Candidate: "Trump", Electors: 10 },
{ State: "Mississippi", Candidate: "Romney", Electors: 6 },
{ State: "Missouri", Candidate: "Romney", Electors: 10 },
{ State: "Montana", Candidate: "Romney", Electors: 3 },
{ State: "Nebraska", Candidate: "Romney", Electors: 5 },
{ State: "Nevada", Candidate: "Trump", Electors: 6 },
{ State: "New Hampshire", Candidate: "Trump", Electors: 4 },
{ State: "New Jersey", Candidate: "Trump", Electors: 14 },
{ State: "New Mexico", Candidate: "Trump", Electors: 5 },
{ State: "New York", Candidate: "Trump", Electors: 29 },
{ State: "North Carolina", Candidate: "Romney", Electors: 15 },
{ State: "North Dakota", Candidate: "Romney", Electors: 3 },
{ State: "Ohio", Candidate: "Trump", Electors: 18 },
{ State: "Oklahoma", Candidate: "Romney", Electors: 7 },
{ State: "Oregon", Candidate: "Trump", Electors: 7 },
{ State: "Pennsylvania", Candidate: "Trump", Electors: 20 },
{ State: "Rhode Island", Candidate: "Trump", Electors: 4 },
{ State: "South Carolina", Candidate: "Romney", Electors: 9 },
{ State: "South Dakota", Candidate: "Romney", Electors: 3 },
{ State: "Tennessee", Candidate: "Romney", Electors: 11 },
{ State: "Texas", Candidate: "Romney", Electors: 38 },
{ State: "Utah", Candidate: "Romney", Electors: 6 },
{ State: "Vermont", Candidate: "Trump", Electors: 3 },
{ State: "Virginia", Candidate: "Trump", Electors: 13 },
{ State: "Washington", Candidate: "Trump", Electors: 12 },
{ State: "West Virginia", Candidate: "Romney", Electors: 5 },
{ State: "Wisconsin", Candidate: "Trump", Electors: 10 },
{ State: "Wyoming", Candidate: "Romney", Electors: 3 }
]
$(function() {
var mapSample = new ej.datavisualization.Map($("#map"), {
layers: [
{
shapeData: usMap,
shapeDataPath: "State",
shapePropertyPath: "name",
dataSource: electionData,
shapeSettings:
{
strokeThickness: 0.5,
autoFill: false,
stroke: "white",

```

```
valuePath: "Electors",  
colorValuePath: "Candidate",  
colorMappings:  
{  
  equalColorMapping:  
  [  
    { value: "Romney", color: "#D84444" },  
    { value: "Trump", color: "#316DB5" }  
  ]  
}  
}]  
});  
});
```



### Color Palette

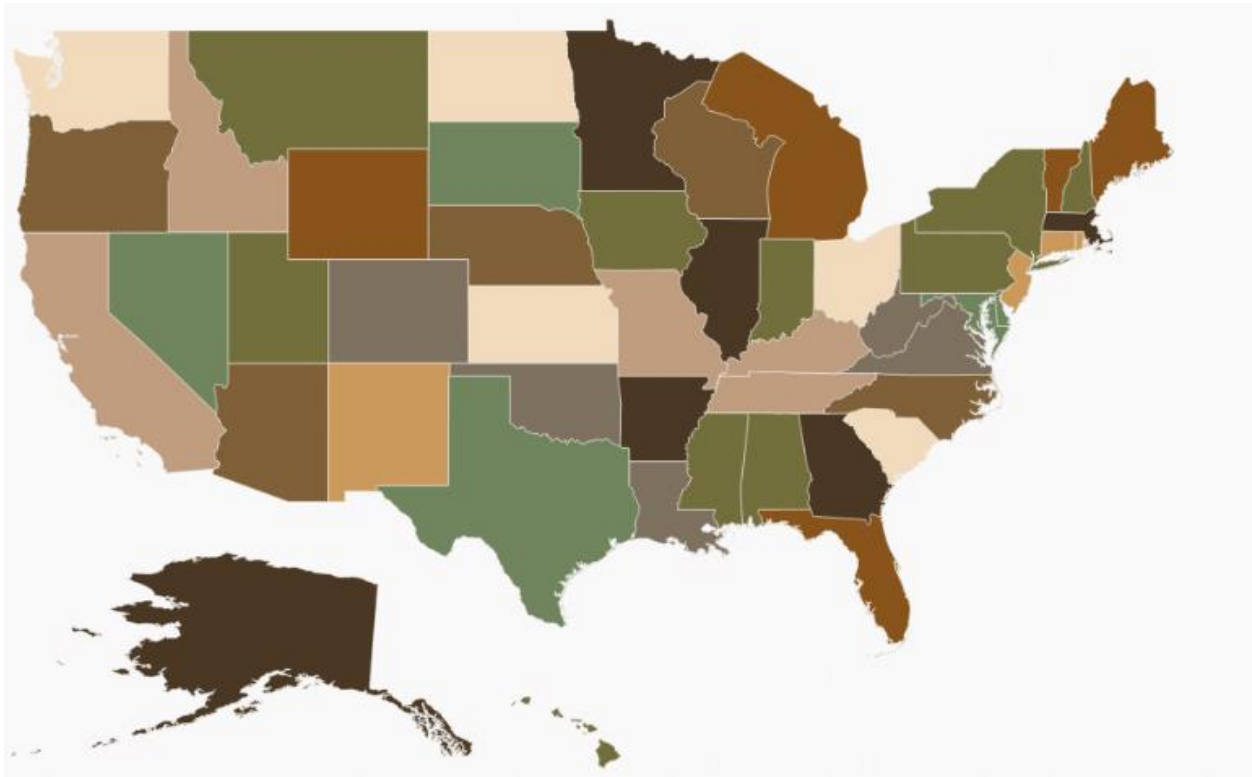
#### AutoFill

When `autoFill` property is set to true, shapes are filled with default colors from built-in palettes or custom palette.

### JAVASCRIPT

```
$(function() {  
  var mapSample = new ej.datavisualization.Map($("#map"), {  
    layers: [  
      {  
        shapeData: usMap,  
        shapeSettings: {  
          strokeThickness: 0.5,
```

```
stroke: "white",  
autoFill: true  
}  
}]  
});  
});
```



### Color Palette

The `colorPalette` property determines whether the auto fill colors are fetched from built-in color palettes or custom palette.

The `colorPalette` property can be set with `palette1`, `palette2`, `palette3` and custom palette values where `palette1`, `palette2` and `palette3` are built-in color palettes and default value for this property is “`palette1`”.

The `customPalette` property is used to set an array of colors to be auto filled in shapes.

This property is enabled only when `colorPalette` property value is set to “`customPalette`”.

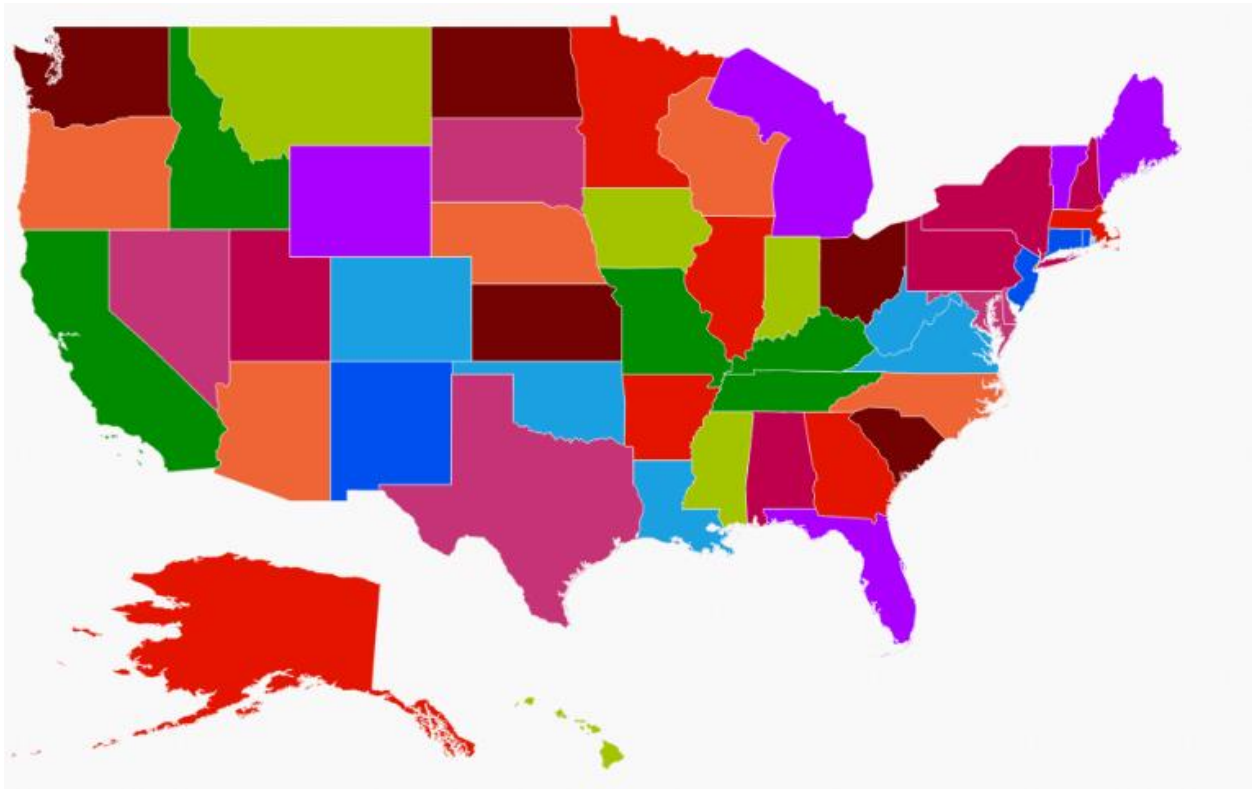
### JAVASCRIPT

```
$(function () {  
  var mapSample = new ej.datavisualization.Map($("#map"), {  
    layers: [  
      {  
        // ...  
        ShapeSettings: {  
          // ...  
          autoFill: true,  
          colorPalette: "customPalette",  
        }  
      }  
    ]  
  });  
});
```

```

customPalette: ["#E51400", "#A4C400", "#730202", "#008B00", "#EF6535",
"#1BA0E2", "#C63477", "#0050EF", "#BF004D", "#AA00FF"]
// ...
}
}]
});
});

```



### Tooltip

The tooltip is displayed only when you set `showTooltip` to “true” in the shape layers. By default, it takes the property of the bound object that is referred in the `valuePath` and displays its content on hovering the corresponding shape.

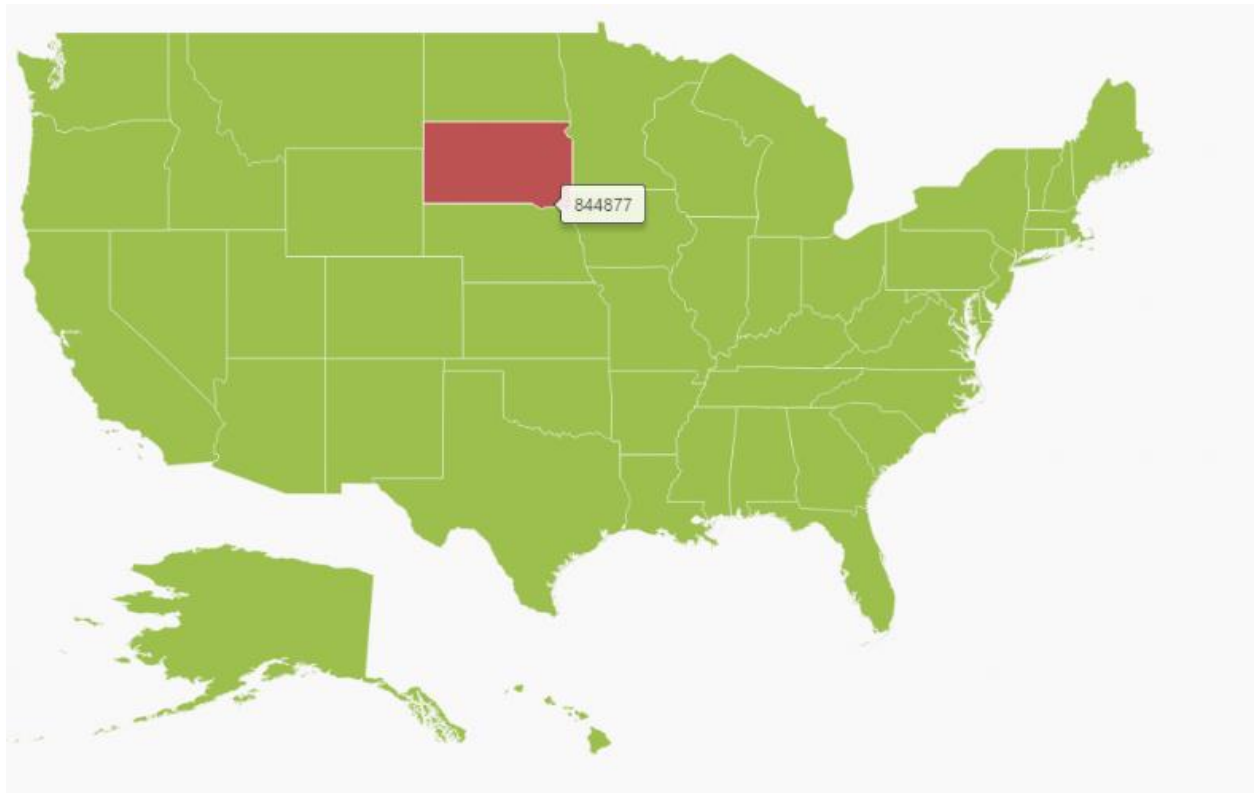
### JAVASCRIPT

```

$(function () {
var mapSample = new ej.datavisualization.Map($("#map"), {
layers: [
{
// ...
ShapeSettings: {
// ...
valuePath: "name",
fill: '#9CBF4E',
strokeThickness: "0.5",
stroke: "White",
highlightStroke: "White",
highlightColor: "#BC5353",
highlightBorderWidth: "1"

```

```
// ...  
}  
showTooltip: true,  
shapeDataPath: "name",  
shapePropertyPath: "name",  
dataSource: USA_State_PopulationData,  
enableMouseHover: true  
}]  
});  
});
```



### Tooltip Template

The `tooltipTemplate` property is used for customizing the template for tooltip.

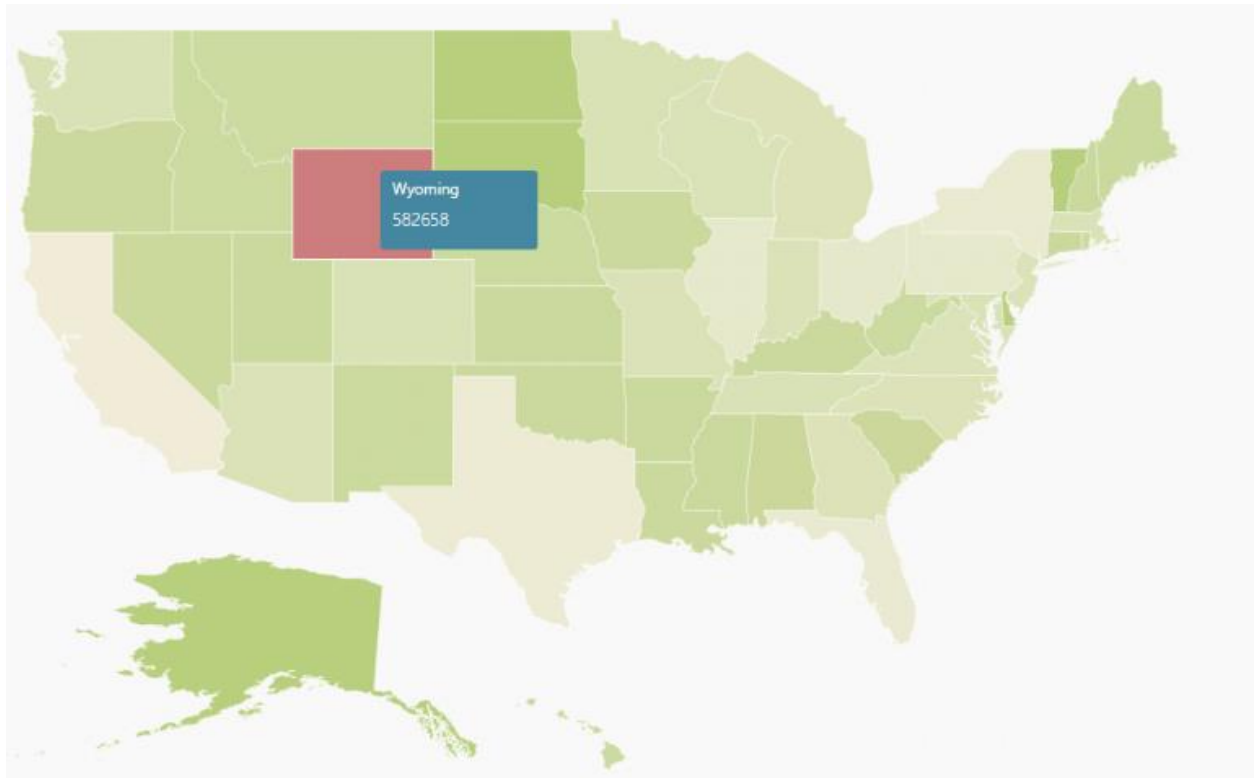
### JAVASCRIPT

```
$(function () {  
    var mapSample = new ej.datavisualization.Map($("#map"), {  
        layers: [  
            {  
                // ...  
                ShapeSettings: {  
                    // ...  
                    valuePath: "name",  
                    fill: '#9CBF4E',  
                    strokeThickness: "0.5",  
                    stroke: "White",  
                    highlightStroke: "White",  
                    highlightColor: "#BC5353",  
                }  
            }  
        ]  
    });  
});
```

```
highlightBorderWidth: "1"
// ...
}
showTooltip: true,
tooltipTemplate: 'myTooltip',
shapeDataPath: "name",
shapePropertyPath: "name",
dataSource: USA_State_PopulationData,
enableMouseHover: true
}]
});
});
<div id="myTooltip" style="display: none;">
<div >
<div style="height:60px;width:120px;background:#4586a0;border-radius:3px;">
<div style="margin-top:-20px;margin-left:9px;padding-top:3px">
<label style="margin-top:-20px;font-weight:normal;font-
size:12px;color:white;font-family:Segoe UI;">{{"{{"}}:name{{}}}}</label>
</div>
<div style="height:10px;"></div>
<div style="margin-top:-10px;margin-left:9px;">
<label style="margin-top:-10px;font-weight:normal;font-
size:14px;color:white;font-family:segoe ui
light;">{{"{{"}}:population{{}}}}</label>
</div>
</div>
</div>
</div>
```

The following screenshot illustrates a map control displaying a Tooltip with template.





#### Customize map background

The Map background can be customized by using the `background` property of the Map.

#### JAVASCRIPT

```
var mapSample = new ej.datavisualization.Map($("#map"), {  
  // ...  
  //Customizing Map background  
  background: "skyblue",  
  // ...  
});
```

#### Base Map Index

Specifies the index of the map to determine the shape layer to be displayed, you can use `baseMapIndex` property and the default value is 0.

#### JS

```
//To set baseMapIndex API value during initialization  
var mapSample = new ej.datavisualization.Map($("#map"), {  
  baseMapIndex:0  
});
```

#### Center Position

Specify the `centerPosition` where map should be displayed

#### JS

```
// Set the centerPosition during initialization.
```

```
var mapSample = new ej.datavisualization.Map($("#map"), {centerPosition: [38.5000, -98]});
```

### Label Settings

The **labelSettings** defines the basic customization settings of labels in the map.

The below properties are used for **labelSettings**

- **enableSmartLabel** - enable or disable the enableSmartLabel property.
- **labelLength** - set the labelLength property.
- **labelPath** - set the labelPath property.
- **showLabels** - The property specifies whether to show labels or not
- **smartLabelSize** - set the smartLabelSize property.

### JAVASCRIPT

```
$(function () {  
var mapSample = new ej.datavisualization.Map($("#map"), {  
layers: [  
{  
shapeData: usMap,  
enableMouseHover: true,  
LabelSettings: {  
enableSmartLabels: false,  
labelLength: "2",  
labelPath: "",  
showLabels: false,  
showLabelSize: "fixed",  
}  
}]  
});  
});
```

### Map Elements

Map control contains a set of map elements such as shapes, bubbles, markers, legend, labels and data items that can be visualized with the customized appearance showing additional information on the map by using the bound data.

### Markers

Markers are notes used to leave some message on the map.

There are two ways to set marker for map.

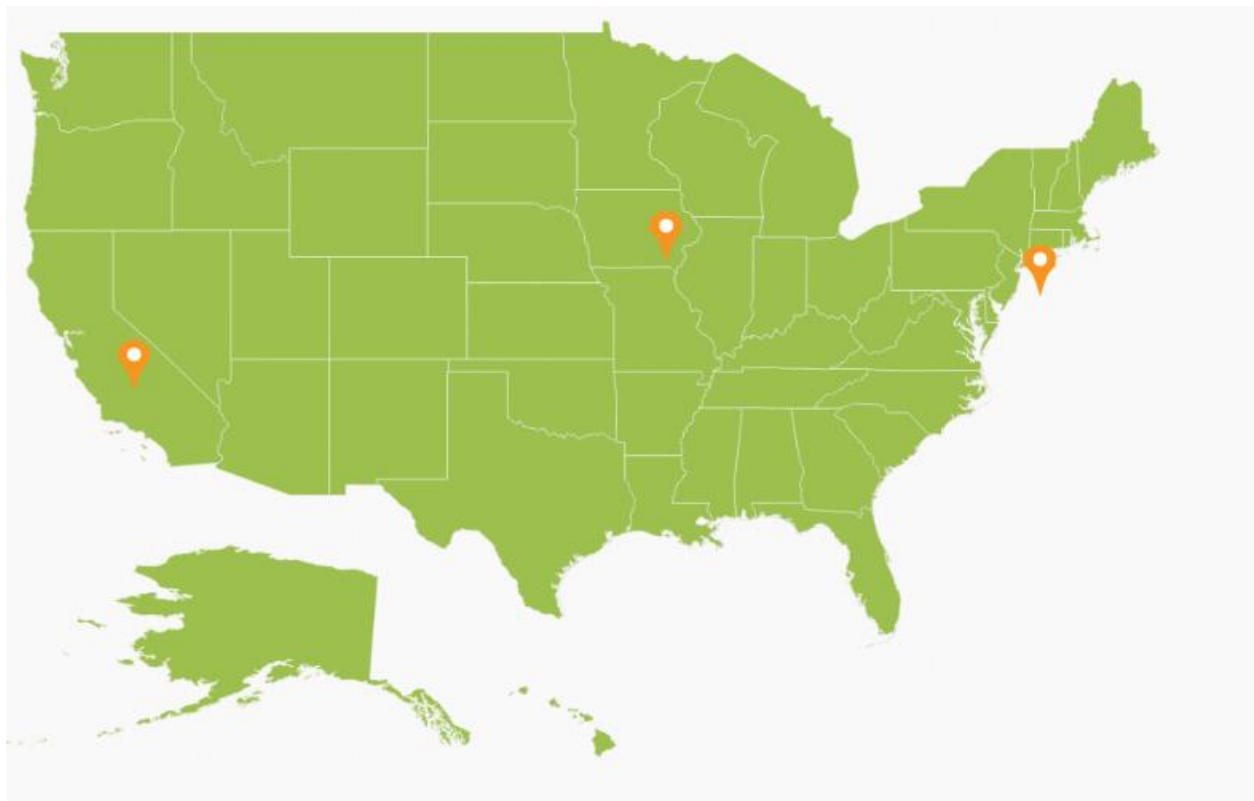
1. Marker and marker template
2. Adding marker objects to map.

### Markers and Marker Template

The **markers** property has a list of objects that contains the data for Annotation. By default, it displays the bound data at the specified latitude and longitude. The **markerTemplate** property is used for customizing the template for markers.

**JAVASCRIPT**

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
var markers = [
{ latitude: 37.0000, longitude: -120.0000, city: "California" },
{ latitude: 40.7127, longitude: -74.0059, city: "New York" },
{ latitude: 42, longitude: -93, city: "Iowa" }
];
module MapComponent {
$(function () {
var mapSample = new ej.datavisualization.Map($("#map"), {
layers: [
{
// ...
markers: markers,
markerTemplate: 'template'
}
}
});
});
}
<div id="template" style="display: none;">
<div>
<div style="background-
image:url(http://js.syncfusion.com/demos/web/Images/map/pin.png);margin-
left:3px;height:40px;width:25px;margin-top:-15px;">
</div>
</div>
</div>
```



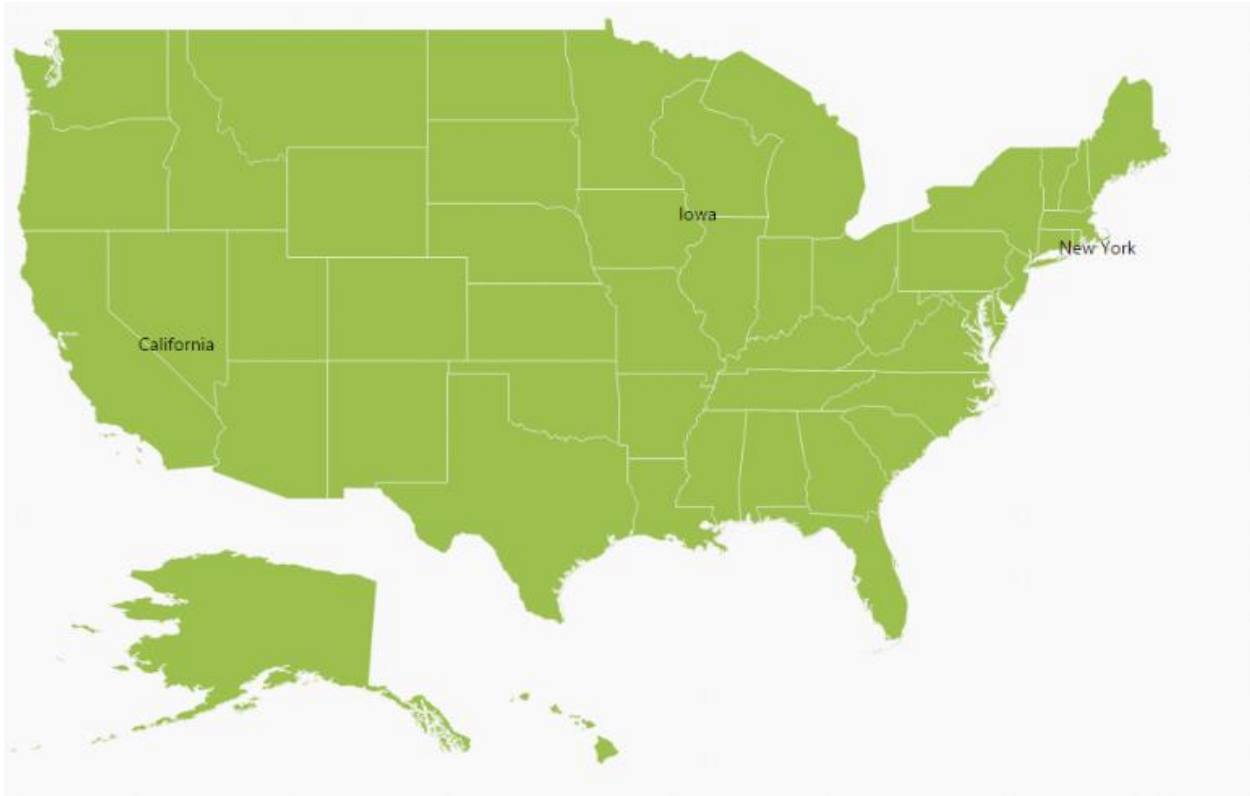
### *Adding Marker objects to the map*

Without datasource, n number of markers can be added to shape layers with `markers` property. Each marker object contains the following list of properties.

- `label` - Text that displays some information about the annotation in text format.
- `latitude` - Latitude point determine the Y-axis position of annotation.
- `longitude` - Longitude point determine the X-axis position of annotation.

### **JAVASCRIPT**

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
var markers = [
{ latitude: 37.0000, longitude: -120.0000, city: "California" },
{ latitude: 40.7127, longitude: -74.0059, city: "New York" },
{ latitude: 42, longitude: -93, city: "Iowa" }
];
module MapComponent {
$(function () {
var mapSample = new ej.datavisualization.Map($("#map"), {
layers: [
{
// ...
markers: markers,
markerTemplate: 'template'
}
}
});
});
}
<div id="template" style="display: none;">
<div>
<div style="margin-left:8px;height:45px;width:120px;margin-top:-23px;">
<label class="label1" style="color:black;margin-left:15px;font-weight:normal">{{:city}}</label>
</div>
</div>
</div>
```



### Bubbles

Bubbles in the **Maps** control represent the underlying data values of the map. Bubbles are scattered throughout the map shapes that contain bound values.

Bubbles are included when data binding and the `bubbleSettings` is set to the shape layers.

### Properties

| Property       | Type                            | Description  |
|----------------|---------------------------------|--|
| maxValue       | String                          | Get or sets the maximum height and width of the bubble.  |
| minValue       | String                          | Gets or sets the minimum height and width of the bubble.   |
| colorValuePath | String                          | Get or sets the field value that is to be fetched from data for each bubble used for determining the bubble color. |
| valuePath      | String                          | Gets or sets the field value that is to be fetched from data for each bubble.                                      |
| colorMappings  | Collection of RangeColorMapping | Gets or sets the tree map colors.  |
| Color          | String                          | Gets or sets the fill color for bubbles.   |
| showTooltip    | Boolean                         | Enable or disable the tooltip for bubbles.   |

|                 |         |   |
|-----------------|---------|---|
| tooltipTemplate | String  | Gets or sets the tooltip template for bubbles.                                    |
| bubbleOpacity   | Boolean | Specifies the <code>bubbleOpacity</code> value of bubbles for shape layer in map. |
| showBubble      | Boolean | Specifies the <code>showBubble</code> visibility status map.                      |

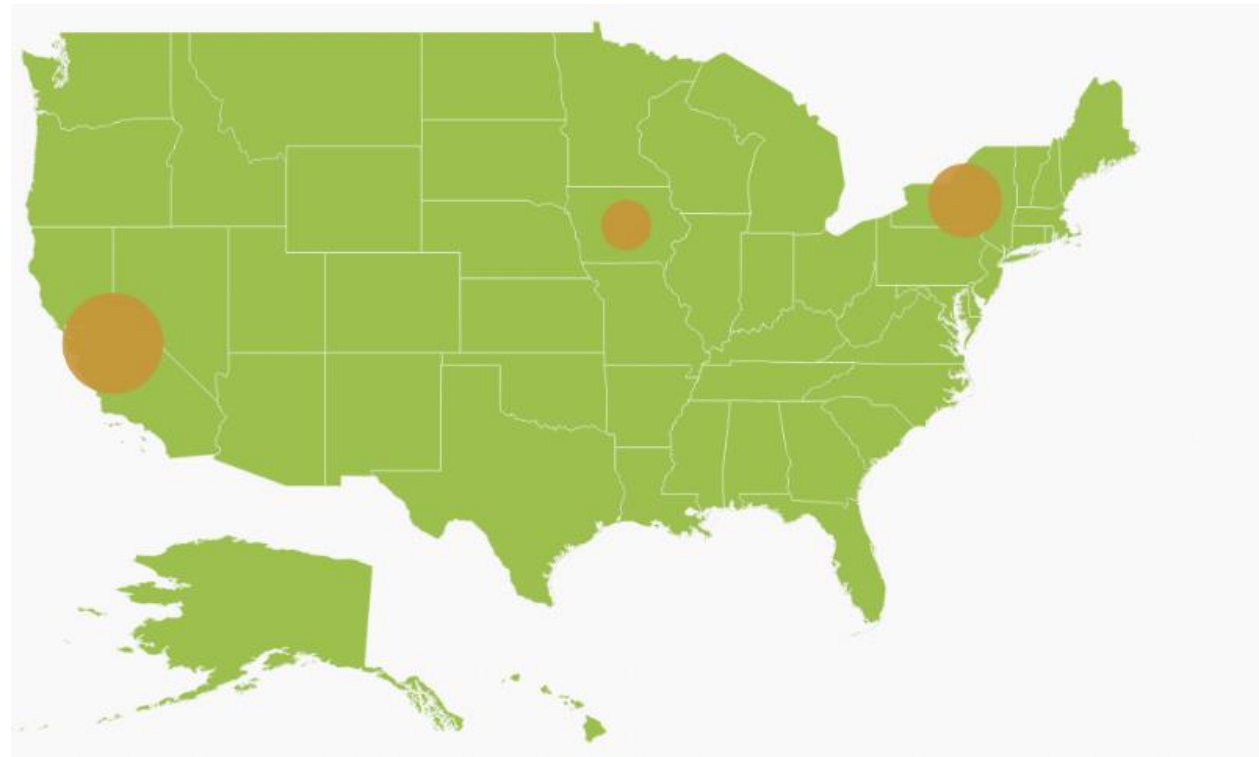
### Add Bubbles to the Map

To add bubbles to a map, the bubble marker setting is added to the shape file layer. Create the Model and ViewModel as illustrated in the Data Binding topic and add the following code. Also set the `maxValue`, `minValue`, and `valuePath` properties as illustrated in the following code sample.

**Note:** `Tooltip` and `Color Mappings` for bubble is to be set as similar to the tooltip and color mappings set in the layers and shapeSettings. For more details, refer to the Tooltip and Color Mappings section.

### JAVASCRIPT

```
$(function () {
    var mapSample = new ej.datavisualization.Map($("#map"), {
        layers: [
            {
                shapeData: usMap,
                shapeDataPath: "name",
                shapePropertyPath: "name",
                dataSource: [
                    { name: "California", population: "38332521" },
                    { name: "New York", population: "19651127" },
                    { name: "Iowa", population: "3090416" }
                ],
                enableMouseHover: true,
                shapeSettings: {
                    fill: "#9CBF4E",
                    strokeThickness: "0.5",
                    stroke: "White"
                },
                bubbleSettings: {
                    showBubble: true,
                    minValue: "20",
                    maxValue: "40",
                    color: "#C99639",
                    valuePath: "population"
                }
            }
        ]
    });
});
```



### Legend

A legend is a key used on a map that contains swatches of symbols with descriptions. It provides valuable information for interpreting what the map is displaying and can be represented in various colors, shapes or other identifiers based on the data. It gives a breakdown of what each symbol represents throughout the map.

### Visibility

The Legends can be made visible by setting the `showLegend` property of `legendSettings` to true.

### Positioning of the Legend

The legend can be positioned in two ways.

1. Absolute Position.
2. Dock Position.

### Absolute Position

Based on the margin values of X and Y-axes, the Map legends can be positioned with the support of `positionX` and `positionY` properties available in `legendSettings`. For positioning the legend based on margins corresponding to a map, `position` value is set as `'none'`.

### Dock Position

The map legends can be positioned in following locations within the container.

1. `topLeft`
2. `topCenter`
3. `topRight`
4. `centerLeft`

5. center
6. centerRight
7. bottomLeft
8. bottomRight
9. bottomCenter
10. bottomRight
11. none

You can set this option by using `dockPosition` property in `legendSettings`.

#### *Legend Size*

The map legend size can be modified by using the `height` and `width` properties in `legendSettings`.

#### *Legend for Shapes*

The Layer shape type legends can be generated for each color mappings in shape settings.

---

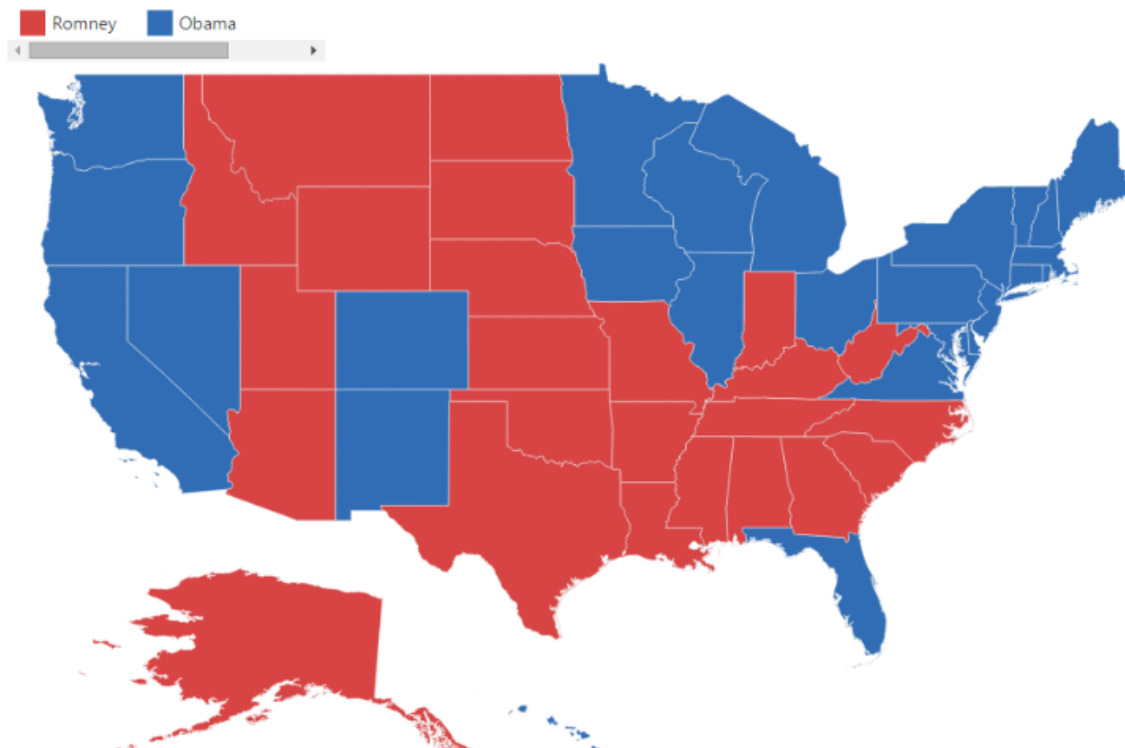
**Note:** Here, Equal Color Mapping code sample for shapeSettings with color mappings is referred.

---

#### **JAVASCRIPT**

```
$(function () {  
  var mapSample = new ej.datavisualization.Map($("#map"), {  
    layers: [  
      {  
        // ...  
        legendSettings:{  
          showLegend:true,  
          position:"bottomleft",  
          height: 30,  
          width: 70,  
        },  
        // ...  
      }]  
    });  
});
```





#### Interactive Legend

The legends can be made interactive with an arrow mark indicating the exact range color in the legend when the mouse hovers over the corresponding shapes. You can enable this option by setting `mode` property in `legendSettings` value as "interactive" and default value of `mode` property is "default" to enable the normal legend.

#### Title for Interactive Legend

You can provide the title for interactive legend by using `title` property in `legendSettings`.

#### Label for Interactive Legend

You can provide the left and right labels to interactive legend by using `leftLabel` and `rightLabel` properties in `legendSettings`.

---

**Note:** Here, Range Color Mapping code snippet for `shapeSettings` with color mappings is referred.

---

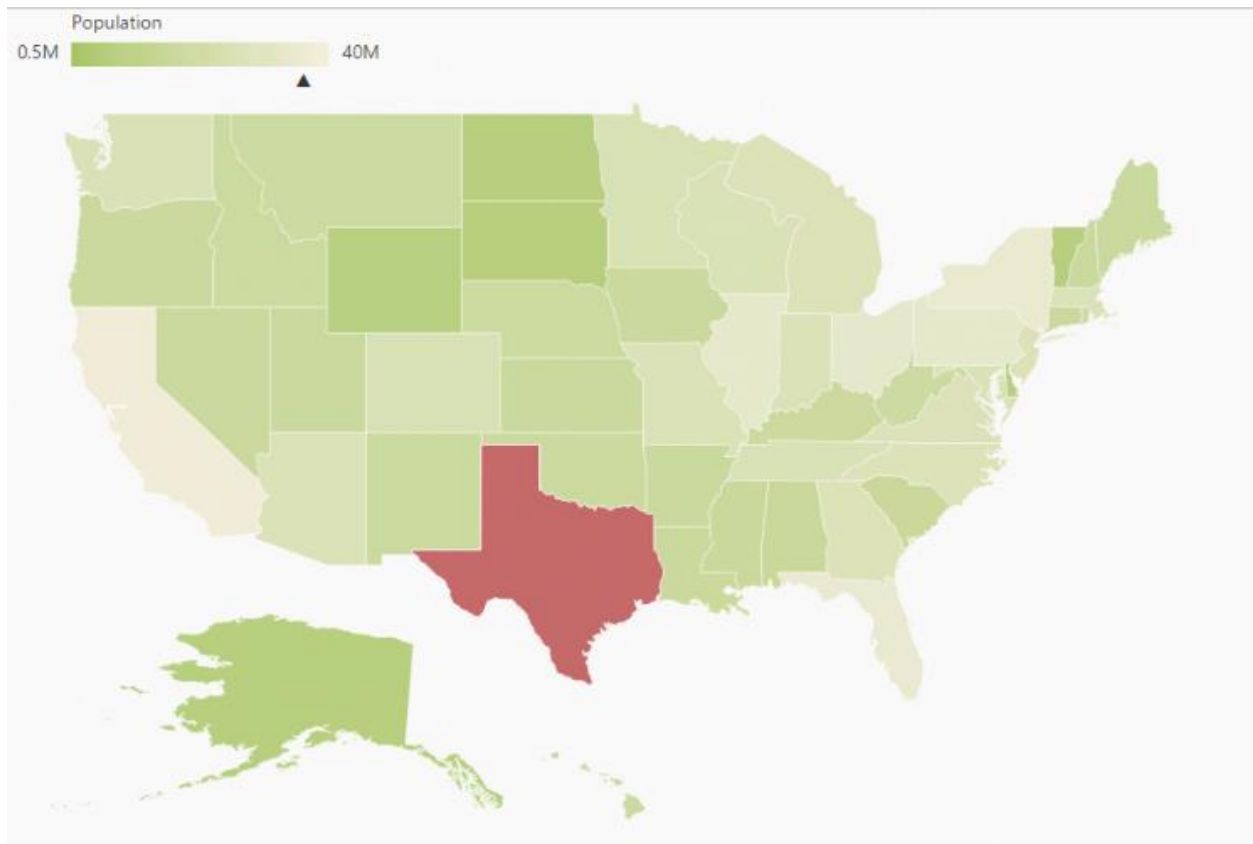
#### JAVASCRIPT

```
$(function () {  
  var mapSample = new ej.datavisualization.Map($("#map"), {  
    layers: [  
      {  
        // ...  
        legendSettings: {  
          showLegend: true,  
          dockOnMap: true,  
          height: 15,  
          width: 150,  
          position: "topleft",  
          mode: "interactive",  
          title: "Population",  
        },  
      },  
    ],  
  });  
});
```

```

leftLabel: "0.5M",
rightLabel: "40M"
},
// ...
}]
});
});

```



### Bubble Legend

A bubble legend feature is used to provide the key (legend) for another map element bubble. You can activate the Bubble legend by setting the enum `type` in `legendSettings` as "bubble" and this enables you to easily identify what value a particular bubble is representing.

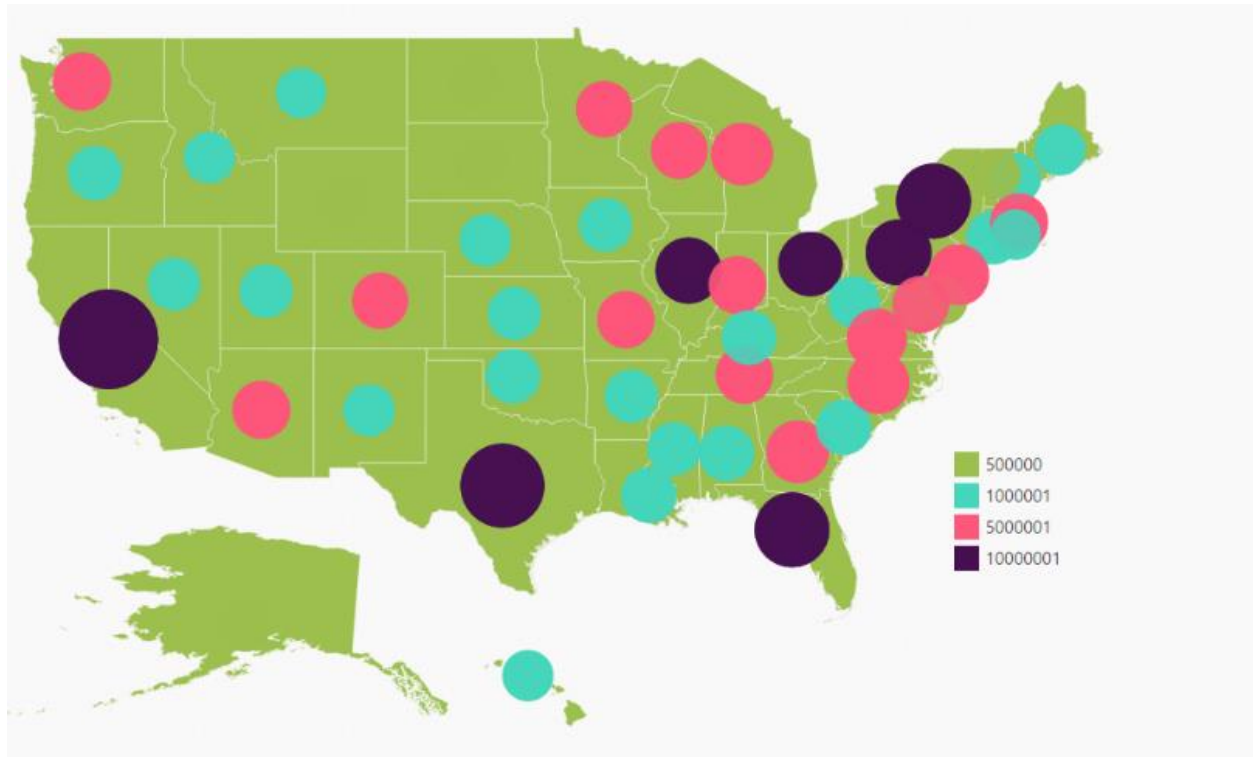
### JAVASCRIPT

```

$(function () {
var mapSample = new ej.datavisualization.Map($("#map"), {
layers: [
{
legendSettings: {
// ...
type: "bubbles",
// ...
},
bubbleSettings: {
showBubble: true,
valuePath: "population",

```

```
minValue: 20,  
maxValue: 40,  
colorMappings:  
{  
  rangeColorMapping:  
  [  
    {  
      from: 500000,  
      to: 1000000,  
      color: "#9CBF4E",  
      range: 10688  
    },  
    {  
      from: 1000001,  
      to: 5000000,  
      color: "#45D6BD",  
      range: 19390  
    },  
    {  
      from: 5000001,  
      to: 10000000,  
      color: "#FF567C",  
      range: 18718  
    },  
    {  
      from: 10000001,  
      to: 40000000,  
      color: "#470F52",  
      range: 30716  
    }  
  ],  
}  
shapeData: usMap  
}  
});  
});
```



## User Interaction

Options like zooming, panning, and map selection enables the effective interaction on Map elements.

### Map Selection

Each shape in the Map can be selected and deselected during interaction with the shapes.

The `selectionColor` property is used to get or set the selected shape color. The `selectionStroke` and `selectionStrokeWidth` properties are used to customize the selected shape border.

You can select the shape by tapping the shape. The Single selection is enabled by the `enableSelection` property of shape layer. When `enableSelection` is set to false, the shapes cannot be selected.

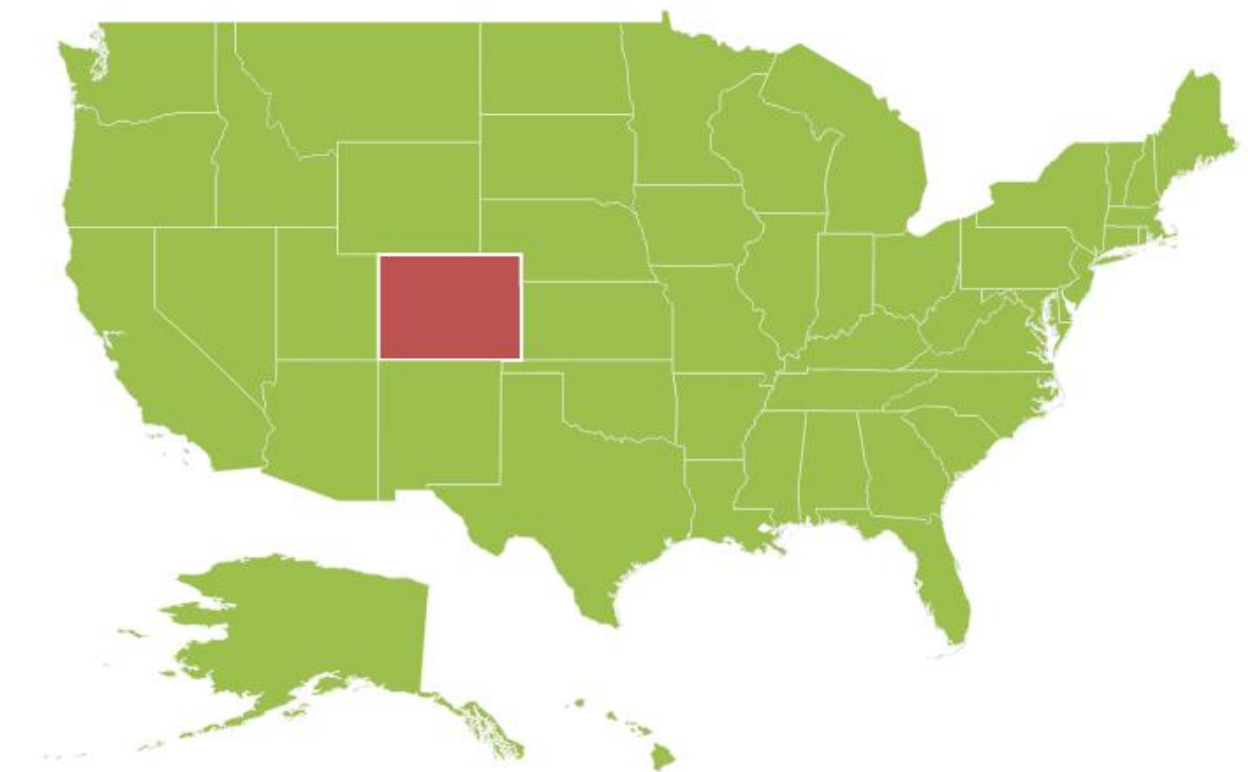
### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module MapComponent {
  $(function() {
    var mapSample = new ej.datavisualization.Map($("#map"), {
      layers: [
        {
          shapeData: usMap,
          shapeSettings: {
            fill: "#9CBF4E",
            strokeThickness: "0.5",
            stroke: "White",
            selectionStrokeWidth: 2,
            selectionStroke: "white",
            selectionColor: "#BC5353"
          },
          enableSelection: true
        }
      ]
    });
  });
}

```

```
}  
]  
});  
});  
}
```

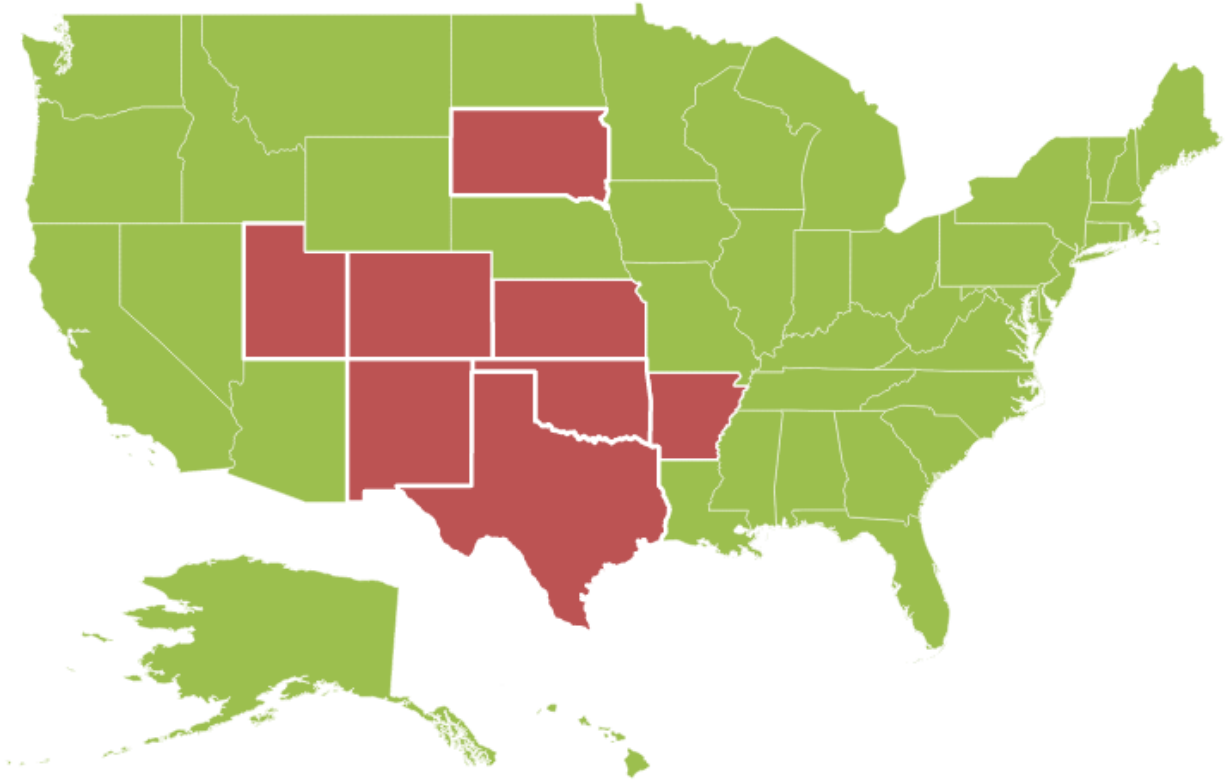


### MultiSelection

This feature enables you to select multiple Map shapes on mouse taps accompanied by the **"Control"** key press. To enable this feature, set the `selectionMode` property as **"multiple"** along with the `enableSelection` property.

### JAVASCRIPT

```
$(function() {  
  var mapSample = new ej.datavisualization.Map($("#map"), {  
    layers: [  
      {  
        shapeData: usMap,  
        // ...  
        enableSelection: true,  
        selectionMode: "multiple",  
        // ..  
      }  
    ]  
  });  
});
```



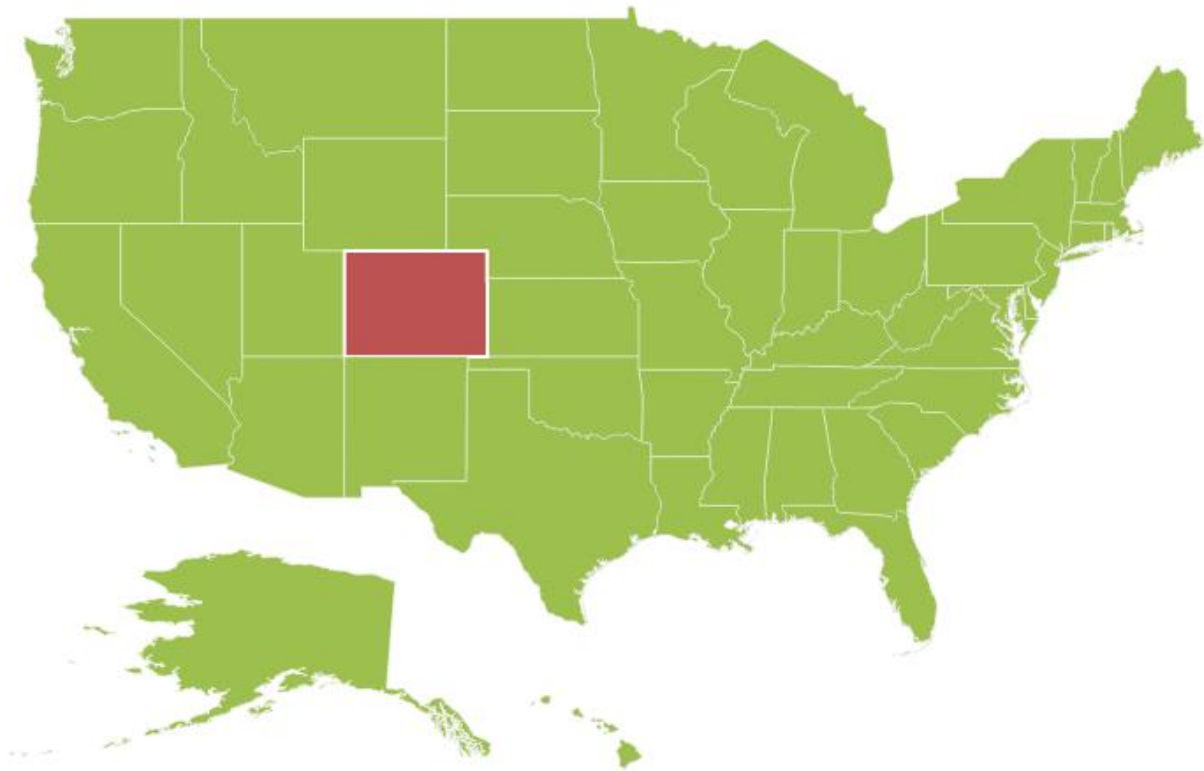
### Dragging On Selection

This feature enables you to select the shapes by dragging over the shapes. While dragging over the shapes, a rectangle is generated and the shapes that comes within the rectangle is selected.

You can enable this feature by setting the `draggingOnSelection` property in the `layers` to `True`.

### JAVASCRIPT

```
$(function() {  
  var mapSample = new ej.datavisualization.Map($("#map"), {  
    layers: [  
      {  
        // ...  
        draggingOnSelection: true  
        // ...  
      }  
    ]  
  });  
});
```



### Zooming

The zooming feature enables you to zoom in and out the Map to show in-depth information. It is controlled by the `level` property of the Map. When the zoom level of the Map control is increased, the Map is zoomed in. When the zoom level is decreased, then the Map is zoomed out.

### Properties

The following properties are related to the zooming feature of the **Maps** control:

1. `level`
2. `enableZoom`
3. `minValue`
4. `maxValue`

### Level

The `level` property determines the Map's scale size when zooming. The default value of `level` is 1.

---

**Note:** `level` cannot be less than 1

---

### EnableZoom

The `enableZoom` property enables or disables the zooming feature.

### MinValue

The `minValue` property is used to set the minimum zoom level of the Map.

### MaxValue

The `maxValue` property is used to set the maximum zoom level of the Map.

## **JAVASCRIPT**

```
$(function() {  
  var mapSample = new ej.datavisualization.Map($("#map"), {  
    layers: [  
      {  
        shapeData: usMap  
      }  
    ],  
    zoomSettings:{  
      enableZoom: true,  
      minValue: 1,  
      maxValue: 20,  
      level: 1  
    }  
  });  
});
```

### Factor

Specifies the zoom factor for map zoom value, you can use **factor** property.

### JAVASCRIPT

```
$(function() {  
  var mapSample = new ej.datavisualization.Map($("#map"), {  
    layers: [  
      {  
        shapeData: usMap  
      }  
    ],  
    zoomSettings:{  
      enableZoom: true,  
      factor: 1  
    }  
  });  
});
```

### Additional Options to Zoom the Map

Maps can be zoomed by using the following options also,

- Zoom method.
- mouse scroll.
- mouse double tap.
- shape selection
- Position

### Zoom method

You can zoom the Maps by using **zoom** method. The **zoom** method contains parameter zoom value. The Map can be zoomed or scaled based on zoom value parameter.

### JAVASCRIPT

```
$("#map").ejMap("zoom", 2);
```

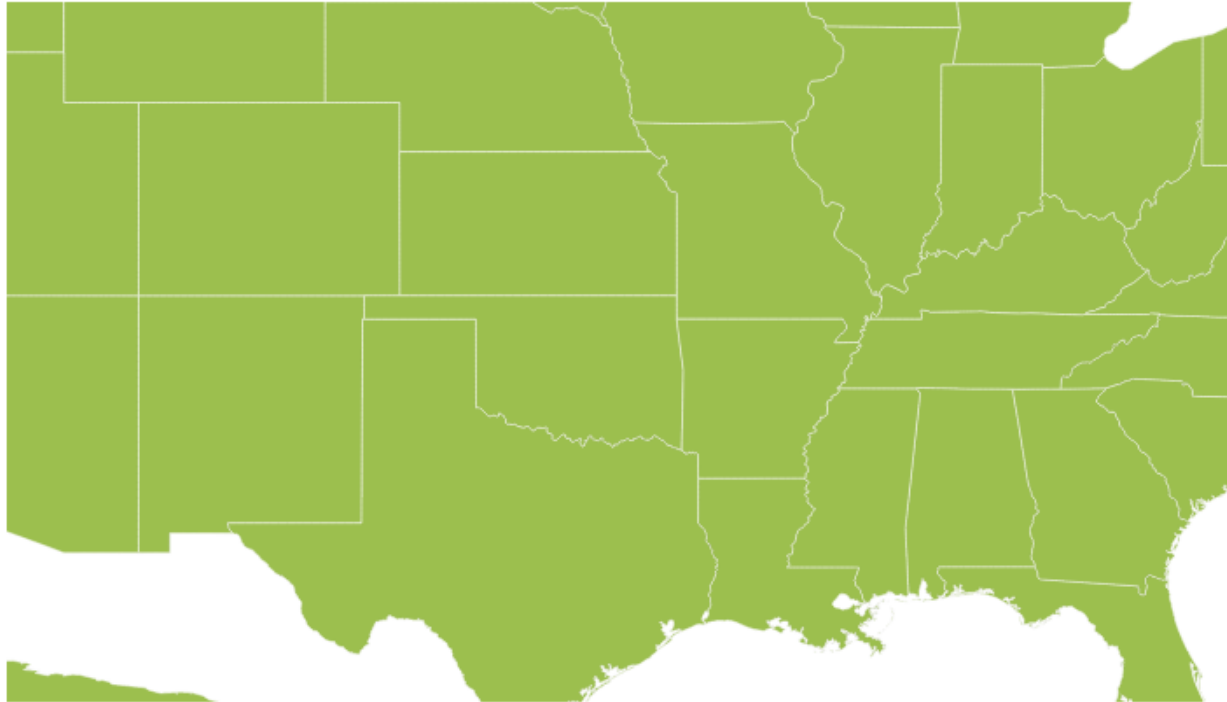


### Mouse scroll

You can zoom the Map with mouse events by using mouse scroll. When the mouse is scrolled up, the Map is zoomed in and when the mouse is scrolled down, the Map is zoomed out.

### Mouse double tap

When the Map is double-tapped by using mouse, the zoom in operation is performed.



### Shape Selection

Map shape is zoomed to the whole Map area on the shape selected. Animation can be applied for that zooming by using the `enableAnimation` property as true.

You can enable this feature by setting `enableZoomOnSelection` property value as 'True'.

When `enableZoomOnSelection` property is set to true, then zooming by double click is muted.

### JAVASCRIPT

```
$(function() {  
  var mapSample = new ej.datavisualization.Map($("#map"), {  
    // ...  
    zoomSettings:  
    {  
      enableZoomOnSelection: true  
    },  
    // ...  
  });  
});
```

### Positioning

Depending on the latitude and longitude, you can zoom the Map to the exact position. All locations are considered as latitude and longitude values and the exact location is considered as Map coordinates.

The `navigateTo` is a method that allows you to zoom the Map control to the given location. This method contains three attributes as follows.

| Attribute | Type   | Description                     |
|-----------|--------|---------------------------------|
| Latitude  | Double | Latitude point of the position  |
| Longitude | Double | Longitude point of the position |
| level     | Double | Zoom level of the map           |

#### JAVASCRIPT

```
function buttonClick() {  
    var mapSample = new ej.datavisualization.Map($("#map"));  
    mapSample.navigateTo(13, 80, 5);  
}
```

#### Panning

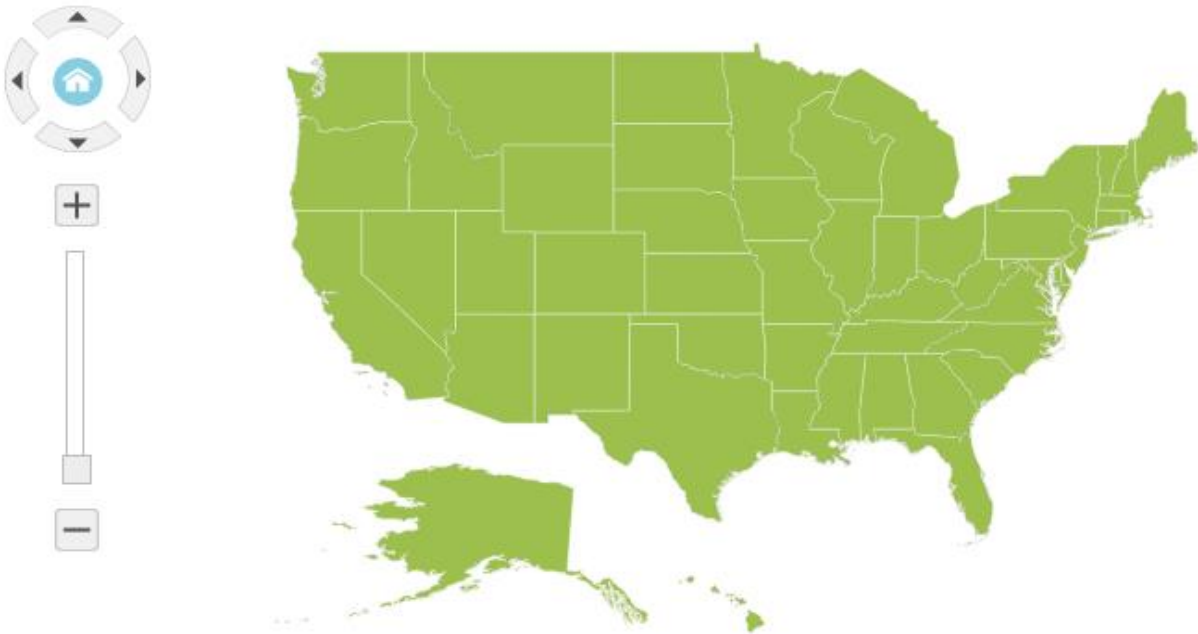
The panning feature enables the Map navigation. The `enablePan` property is used to enable or disable the panning support.

#### JAVASCRIPT

```
$(function() {  
    var mapSample = new ej.datavisualization.Map($("#map"), {  
        // ...  
        enablePan: true  
        // ...  
    });  
});
```

#### Navigation Control

**Navigation** control is built-in with **Maps** control. With Navigation control, **Maps** can be panned in any direction and zoomed. It is possible to show or hide the NavigationControl by `enableNavigation` property.



## JAVASCRIPT

```
$(function() {  
  var mapSample = new ej.datavisualization.Map($("#map"), {  
    // ...  
    navigationControl:{enableNavigation:true},  
    // ...  
  });  
});
```

### Positions

The Navigation control can be positioned in two ways.

- Absolute Position
- Dock Position

#### Absolute Position

Based on the margin values of X and Y-axes, the navigation control can be positioned with the help of the **x** and **y** properties available in **absolutePosition**. For positioning the navigation control based on margins corresponding to a Map, **dockPosition** value is set as 'none'.

#### Dock Position

The navigation control can be positioned in the following locations within the container.

- topLeft
- topCenter
- topRight
- centerLeft
- center
- centerRight

- bottomLeft
- bottomRight
- bottomCenter
- bottomRight
- none

You can set this option by using `dockPosition` property in `navigationControl`.

#### JAVASCRIPT

```
$(function() {
var mapSample = new ej.datavisualization.Map($("#map"), {
// ...
navigationControl:{enableNavigation:true,orientation:'vertical',absolutePosition:{x:5,y:16},dockPosition:'none'},
// ...
});
});
```

#### Orientation

Set the `orientation` value for navigation control.

| Name       | Description                       |
|------------|-----------------------------------|
| horizontal | specifies the horizontal position |
| vertical   | specifies the vertical position   |

#### JAVASCRIPT

```
//To set orientation value during initialization
var mapSample = new ej.datavisualization.Map($("#map"),
{orientation:'vertical'});
```

#### Content

Specifies the navigation control template for map, you can use `content` property.

#### JAVASCRIPT

```
//To set navigation control template for map during initialization
var mapSample = new ej.datavisualization.Map($("#map"),
{navigationControl:{content:null}});
```

#### Animation

**Animation** is enabled or disabled using `enable animation` property.

#### JAVASCRIPT

```
var mapSample = new ej.datavisualization.Map($("#map"), {
enableAnimation: true,
});
```

### Enable Layer Change Animation

Enables or Disables the animation for layer change in map, you can use `enableLayerChangeAnimation` property and the default value is false.

#### JAVASCRIPT

```
//To set enableLayerChangeAnimation API value during initialization
var mapSample = new ej.datavisualization.Map($("#map"),
{enableLayerChangeAnimation:true});
```

### Responsiveness during browser resize

**Map** is made responsive when resizing the browser by using `isResponsive` property.

#### JAVASCRIPT

```
var mapSample = new ej.datavisualization.Map($("#map"), {
isResponsive: true,
});
```

## Layers

Map is maintained through `layers` and it can accommodate one or more layers.

### Multilayer

The Multilayer support allows you to load multiple shape files in a single container, enabling maps to display more information.

### Adding Multiple Layers in the Map

The shape layers is the core layer of the map. The multiple layers can be added in the shape layers as `subLayers` within the shape layers.

### SubLayer

The subLayer is the collection of shape layers.

In this example, World Map shape is used as shape data by utilizing the “**WorldMap.json**” file in the following folder structure obtained from downloaded Maps\_GeoJSON folder.

```
..\Maps_GeoJSON\
```

You can assign the complete contents in “**WorldMap.json**” file to new JSON object. For better understanding, a JS file “**WorldMap.js**” is already created to store JSON data in JSON object “usMap”

[usa.js]

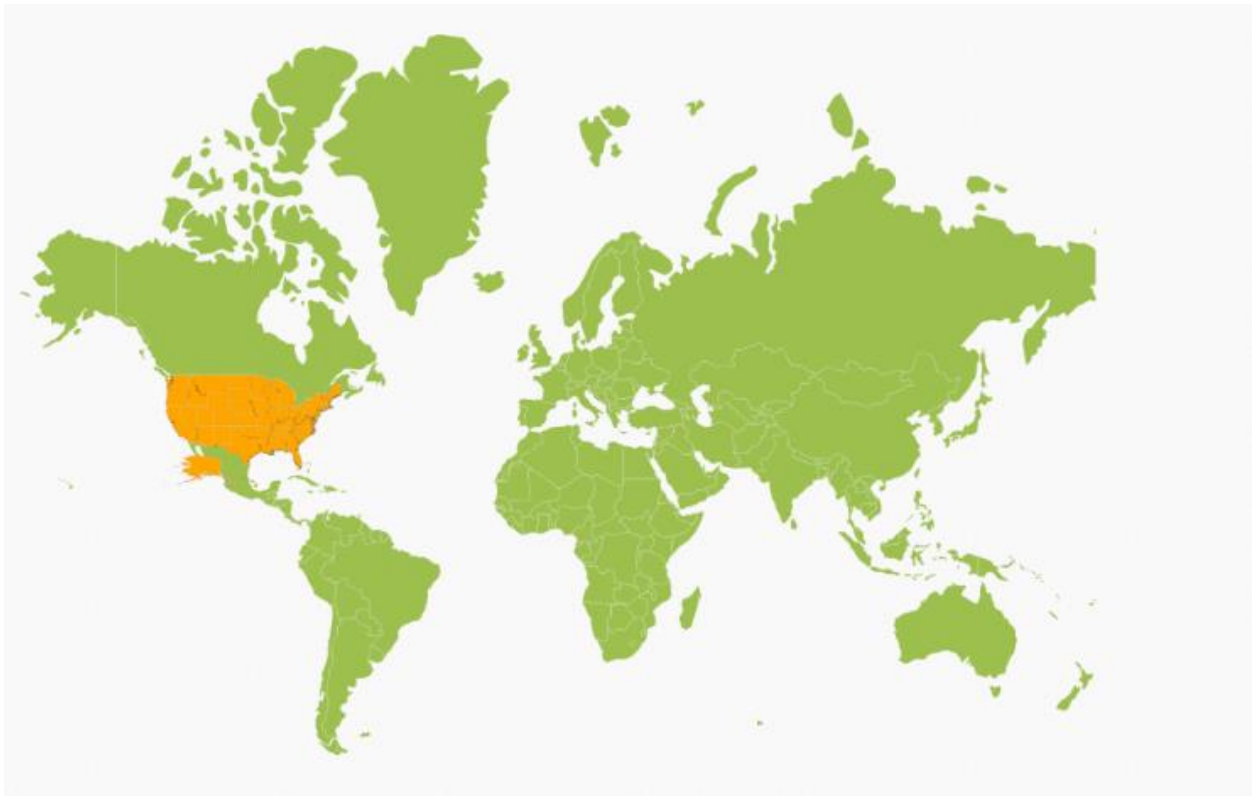
#### JAVASCRIPT

```
var world_map = //Paste all the content copied from the JSON file//
```

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module MapComponentnet {
$(function () {
var mapSample = new ej.datavisualization.Map($("#map"), {
layers: [
```

```
{
  shapeData: world_map,
  shapeSettings: {
    fill: "#9CBF4E",
    strokeThickness: "0.5",
    stroke: "White"
  },
  subLayers: [{
    shapeData: usMap,
    shapeSettings: {
      fill: "orange",
      strokeThickness: "1",
      stroke: "White"
    }
  }]
}
});
});
}
```



## Map Providers

**Map** control support map providers such as OpenStreetMap that can be added to any layers in maps.

### Open Street Map

OpenStreetMap is a map of the entire world. The OpenStreetMap allows you to view, edit and use geographical data in a collaborative way from any place on the Earth.

### Enable OSM

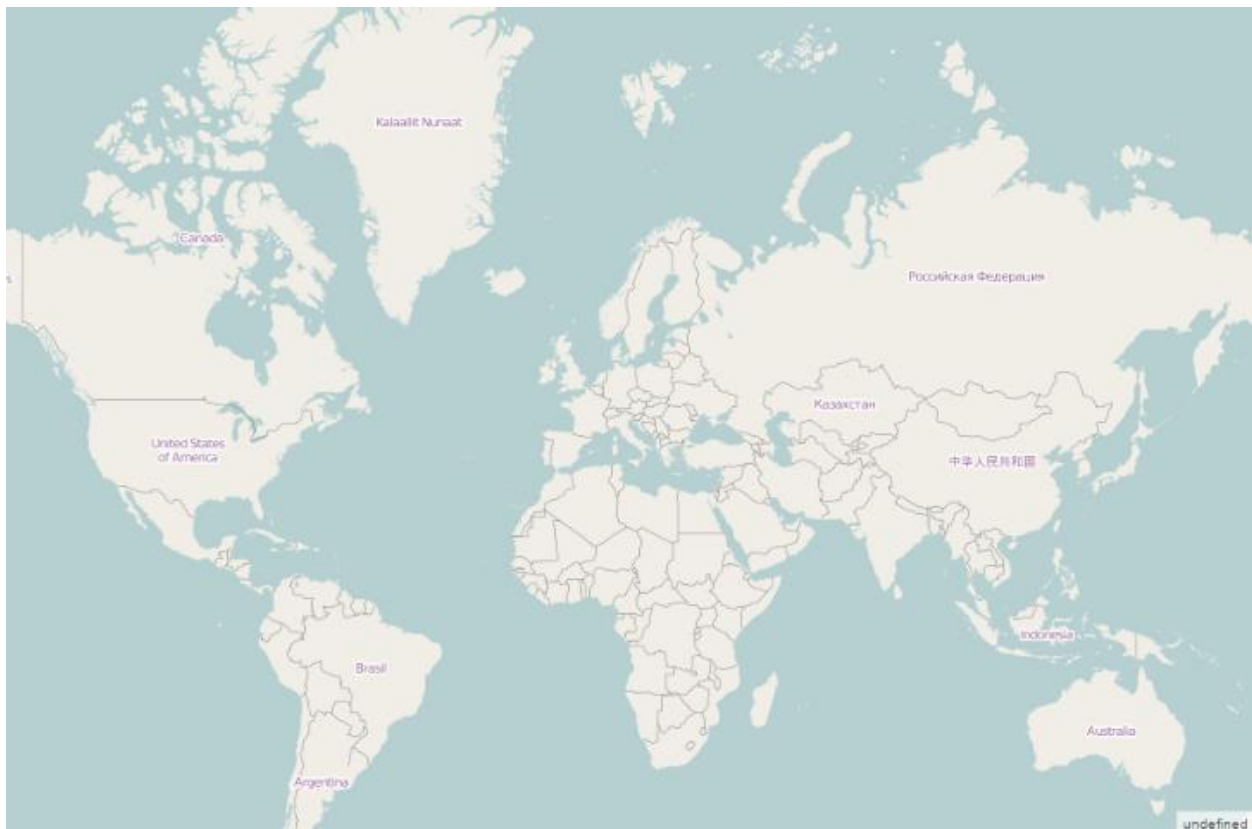
You can enable this feature by setting the `layerType` property value as "OSM".

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module MapComponent {
  $(function () {
    var mapSample = new ej.datavisualization.Map($("#map"), {
      layers: [{
        layerType: 'osm',
        urlTemplate: 'http://a.tile.openstreetmap.org/level/tileX/tileY.png'
      }]
    });
  });
}
```

### URL Template

The `urlTemplate` property determines the format of tile map. You can specify the template for the tile layer.



### Bing Map

Bing Map is a key feature in accessing the external geospatial imagery services for deep-zoom satellite view.

### Enable Bing Maps

You can enable this feature by defining the `layerType` as "bing".

#### JAVASCRIPT

```
var mapSample = new ej.datavisualization.Map($("#map"), {  
  layers: [{  
    layerType: 'bing',  
    bingMapType: "AerialWithLabel",  
    key: '// ...bingMapKey'  
  }]  
});
```

### Key

The bing Map key is provided as input to this `key` property. The Bing Map key can be obtained from <http://www.microsoft.com/maps/create-a-bing-maps-key.aspx>.



### AngularJS Support

**AngularJS** is a **JavaScript** Framework added to a **HTML** page with a `<script>` tag. It extends **HTML** attributes with directives and binds data to **HTML** with expressions. **AngularJS** directives allow you to specify custom and reusable **HTML** tags that moderate the behavior of certain elements. **Angular binding** uses directives to plug its action into the page. Directives, all prefaced with `ng-`, are placed in **HTML** attributes. To know more about **Angular binding** refer

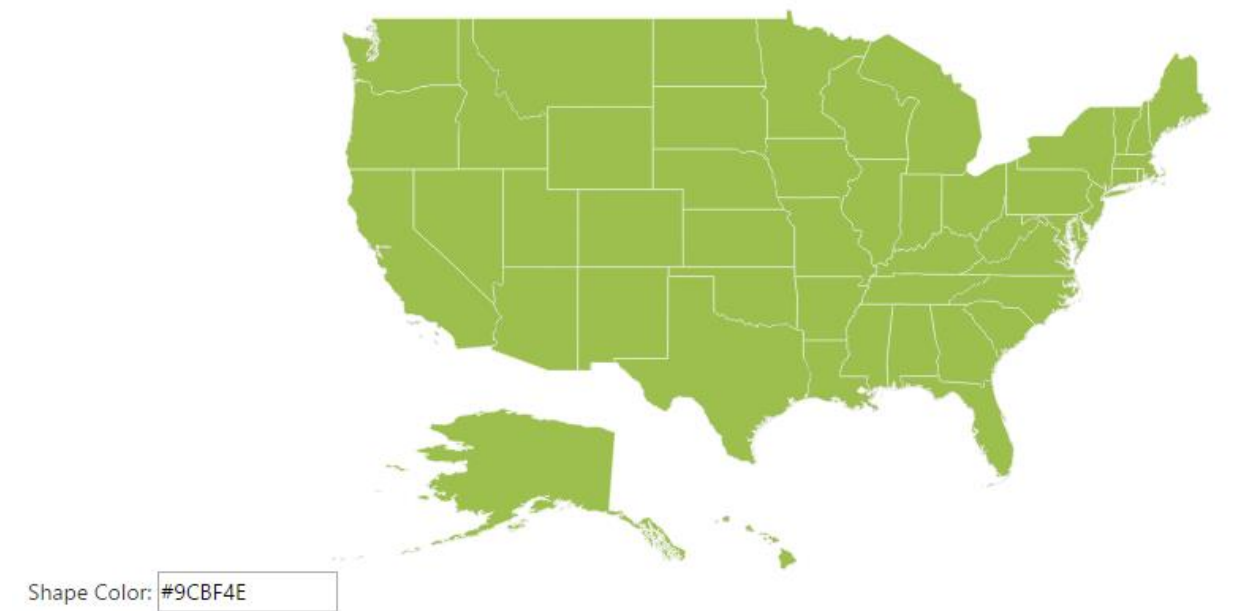
to: <https://help.syncfusion.com/js/angularjs>



Apply the plugin and property assigning the Map element through the directive that starts with the letter “e-”. The following code illustrates how to bind data to the Map component through Angular support.

### HTML

```
//References to be added for AngularJS support.
<script src="
https://ajax.googleapis.com/ajax/libs/angularjs/1.0.1/angular.min.js"></scrip
t>
<script src=" http://cdn.syncfusion.com/js/ej.widget.angular-
latest.min.js"></script>
//Initializes controller
<body ng-controller="Map">
//Initializes Map
<div id="map" style="height: inherit; min-height: 356px;" ej-map e-
zoomsettings- enablezoom="enableZoom">
<div e-layers>
<div e-layer e-shapedata="shapeData" e-shapesettings-fill="fill" e-
shapesettings-strokeThickness="strokeThickness" e-shapesettings-
stroke="stroke">
</div>
</div>
</div>
//Renders a textbox to change the color value
<div>
Shape Color: <input type="text" id="Text11" ng-model="fill" style="width:
110px">
</div>
<script>
angular.module('syncApp', ['ejangular'])
.controller('Map', function ($scope) {
$scope.enableZoom = true,
$scope.shapeData = usMap;
$scope.fill = "#9CBF4E";
$scope.strokeThickness = "0.5";
$scope.stroke = "White";
});
</script>
```



## Methods

*[navigateTo\(latitude, longitude, level\)](#)*

Method for navigating to specific shape based on latitude, longitude and zoom level.

## HTML

```
<div id="map"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MapComponent {
  $(function () {
    var sample = new ej.datavisualization.Map($("#map"), {
    ///...///
    });
    sample.navigateTo();
  });
}
```

*[pan\(direction\)](#)*

Method to perform map panning

## HTML

```
<div id="map"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```

module MapComponent {
  $(function () {
    var sample = new ej.datavisualization.Map($("#map"), {
      //...//
    });
    sample.pan();
  });
}

```

### *refresh()*

Method to reload the map.

### HTML

```
<div id="map"></div>
```

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MapComponent {
  $(function () {
    var sample = new ej.datavisualization.Map($("#map"), {
      //...//
    });
    sample.refresh();
  });
}

```

### *refreshLayers()*

Method to reload the shapeLayers with updated values

### HTML

```
<div id="map"></div>
```

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MapComponent {
  $(function () {
    var sample = new ej.datavisualization.Map($("#map"), {
      //...//
    });
    sample.refreshLayers();
  });
}

```

### *refreshNavigationControl(navigation)*

Method to reload the navigation control with updated values.

**HTML**

```
<div id="map"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MapComponent {
$(function () {
var sample = new ej.datavisualization.Map($("#map"), {
///...//
});
sample.refreshNavigationControl();
});
});
}
```

*zoom(level, isAnimate)*

Method to perform map zooming.

**HTML**

```
<div id="map"></div>
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MapComponent {
$(function () {
var sample = new ej.datavisualization.Map($("#map"), {
///...//
});
sample.zoom();
});
});
}
```

**Events**

*markerSelected*

Triggered on selecting the map markers.

**JAVASCRIPT**

```
<script>
//markerSelected event for Map
$(function () {
var sample = new ej.datavisualization.Map($("#map"), {
markerSelected: function () {
///...//
}
});
});
</script>
```

*mouseleave*

Triggers while leaving the hovered map shape

**JAVASCRIPT**

```
<script>
//mouseleave event for Map
$(function () {
  var sample = new ej.datavisualization.Map($("#map"), {
    mouseLeave: function () {
      //...//
    }
  });
});
</script>
```

*mouseover*

Triggers while hovering the map shape.

**JAVASCRIPT**

```
<script>
//mouseover event for Map
$(function () {
  var sample = new ej.datavisualization.Map($("#map"), {
    mouseover: function () {
      //...//
    }
  });
});
</script>
```

*onRenderComplete*

Triggers once map render completed.

**JAVASCRIPT**

```
<script>
//onRenderComplete event for Map
$(function () {
  var sample = new ej.datavisualization.Map($("#map"), {
    onRenderComplete: function () {
      //...//
    }
  });
});
</script>
```

*panned*

Triggers when map panning ends.

**JAVASCRIPT**

```
<script>
```

```
//panned event for Map
$(function () {
  var sample = new ej.datavisualization.Map($("#map"), {
    panned: function () {
      //...//
    }
  });
});
</script>
```

### *shapeSelected*

Triggered on selecting the map shapes.

#### **JAVASCRIPT**

```
<script>
//shapeSelected event for Map
$(function () {
  var sample = new ej.datavisualization.Map($("#map"), {
    shapeSelected: function () {
      //...//
    }
  });
});
</script>
```

### *zoomedIn*

Triggered when map is zoomed-in.

#### **JAVASCRIPT**

```
<script>
//zoomedIn event for Map
$(function () {
  var sample = new ej.datavisualization.Map($("#map"), {
    zoomedIn: function () {
      //...//
    }
  });
});
</script>
```

### *zoomedOut*

Triggers when map is zoomed out.

#### **JAVASCRIPT**

```
<script>
//zoomedOut event for Map
$(function () {
  var sample = new ej.datavisualization.Map($("#map"), {
    zoomedOut: function () {
      //...//
    }
  });
});
```

```
});  
</script>
```

<a class="" href="http://www.syncfusion.com/copyright" target="\_blank">Copyright &#169; 2001 - 2015 Syncfusion Inc. All Rights Reserved</a>

## MaskEdit

### Overview

The **MaskEdit** control provides an easy and reliable way of collecting user input and displaying standard data in a specific format. Some common uses of the **MaskEdit** control are IP address editors, phone number editors, and Social Security number editors.

### Key Features

- **Mask:** The format in which the value should be entered in the text box can be set using this property.
- **Input Mode:** This feature allows the **MaskEdit** text box to act as an input text box or password text box.
- **Watermark Text:** The **MaskEdit** text box supports watermarking. A watermark is background text that appears in the text box without interfering with user text entry or the readability of the text entered. It can be used to display a ready instruction or other important information for the user. It appears automatically before the text is entered and disappears once the user begins entering text.

### Getting Started

Using the following steps, you can create a **Typescript** MaskEdit component.

#### Creating an MaskEdit in TypeScript

You can create a **Typescript** application with the help of the given [Typescript Getting Started Documentation](#).

Within an index.html file and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>  
<html>  
<head>  
<title>TypeScript Application</title>  
<link  
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/flat-  
azure/ej.web.all.min.css" rel="stylesheet" />  
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>  
<script  
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.web  
.all.min.js" type="text/javascript"></script>  
</head>  
<body>  
<!--Add MaskEdit sample here-->  
</body>  
</html>
```

The can be created from a `input` element with the HTML `id` attribute and pre-defined options set to it.

### HTML

```
<input id="maskEdit" type="text" />
<script src="app.js"></script>
```


- Create `app.ts` file and use the below content

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var mask = new ej.MaskEdit($("#maskEdit"), {
    name: "mask",
    value: "4242422424",
    maskFormat: "99 999-99999",
    width: "100%"
  })
}
```

- Now build your application, so that the **app.ts** file will compiled and automatically generated the **app.js** file which is added to your project (User have nothing to do with this file). Now, whatever code changes that you make in **app.ts** file will be reflected in `app.js` file by compiling build the application.

Execution of above code will render the following output.



### Error Visibility


- The MaskEdit has an option that shows the error value with red colored text. It is used to validate the Mask Edit value. You can set the `showError` property as `"true"` to enable this option.

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var mask = new ej.MaskEdit($("#maskEdit"), {
    name: "mask",
    value: "123456",
    maskFormat: "99 999-99999",
    width: "100%",
    showError: true
  })
}
```



- Execution of above code will render the following output



### CustomCharacter

The MaskEdit allows you to use the custom character option. The specified character is only allowed to enter in the Mask Edit Textbox by using the customCharacter property.

### JS

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
var mask = new ej.MaskEdit($("#maskEdit"), {
  name: "mask",
  value: "4242424",
  maskFormat: "9C 9C9-9C",
  width: "100%",
  customCharacter: "$"
})

```

## Behavior Settings

### Persistence Support

The MaskEdit control provides state maintenance support. You can maintain the previous changes made in the control after a page loads. By default, the **'enablePersistence'** property is set to **'false'** in the **MaskEdit**.

The following steps explain the **State Maintenance** in the **MaskEdit** control.

In the **HTML** page, add a **<div>** element to render the **MaskEdit** widget.

### HTML

```

<input id="maskEdit" type="text" />

```

### JAVASCRIPT

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var mask = new ej.MaskEdit($("#maskEdit"), {
    name: "mask",
    inputMode: ej.InputMode.Text,
    maskFormat: "99-999-99999",
    enablePersistence: true
  })
}

```

Output of MaskEdit with **enablePersistence** is as follows.



### Enabled or Disabled

MaskEdit has an option to enable or disable its element. You can set the **enabled** property as “false” to disable the MaskEdit controls.

The following steps explain the **enabled** property in the **MaskEdit** control.

In the **HTML** page, add a **<div>** element to render the **MaskEdit** widget.

### HTML

```
<input id="maskEdit" type="text" />
```

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var mask = new ej.MaskEdit($("#maskEdit"), {
    name: "mask",
    inputMode: ej.InputMode.Text,
    maskFormat: "99-999-99999",
    enabled: false
  })
}
```

Output when **enabled** is “false” and when **enabled** is “true”.



MaskEdit with disabled state



MaskEdit with default state

### Adjusting MaskEdit Size

**MaskEdit** size can be modified by using the **height** and **width** properties. You can customize the size of **MaskEdit** by using these properties.

The following steps explain the **width** and **height** property in the **MaskEdit** control.

In the **HTML** page, add a **<div>** element to render the **MaskEdit** widget.

### HTML

```
<input id="maskEdit" type="text" />
```

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var mask = new ej.MaskEdit($("#maskEdit"), {
    name: "mask",
    inputMode: ej.InputMode.Text,
    maskFormat: "99-999-99999",
    width: 150,
    height: 50
  })
}
```

Output of MaskEdit after setting “height” and “width” is as follows.



### Define Value

The value of **MaskEdit** can be assigned by using the **value** property. The default value for **value** property is null. Specify the [name](#) attribute value for the mask edit textbox.

You can get the raw value of **MaskEdit** without literals and prompt characters by using the [get StrippedValue](#) method.

Also you get the value of **MaskEdit** with the masked format by using the [get UnstrippedValue](#) method.

The following steps explain the **value** property in the **MaskEdit** control.

In the **HTML** page, add a **<div>** element to render the **MaskEdit** widget.

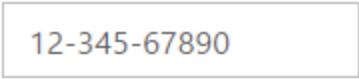
### HTML

```
<input id="maskEdit" type="text" />
```

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var mask = new ej.MaskEdit($("#maskEdit"), {
    name: "mask",
    inputMode: ej.InputMode.Text,
    maskFormat: "99-999-99999",
    value: "1234567890"
  })
}
```

Output of MaskEdit with the **value** property is as follows.



### Read Only Support

**MaskEdit** supports read only option. When you enable the **readOnly** property to the control, the value cannot be changed in the **MaskEdit**. You can set the **readOnly** property as “true” to enable this option.

The following steps explain the **readOnly** property in the **MaskEdit** control.

In the **HTML** page, add a **<div>** element to render the **MaskEdit** widget.

#### HTML

```
<input id="maskEdit" type="text" />
```

#### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var mask = new ej.MaskEdit($("#maskEdit"), {
    name: "mask",
    inputMode: ej.InputMode.Text,
    maskFormat: "99-999-99999",
    value: "123456",
    readOnly: true
  })
}
```

Output of **MaskEdit** when **readOnly** is “true” is as follows. MaskEdit values cannot be edited or changed.



### MaskEdit Properties

#### HidePromptOnLeave

The **MaskEdit** provides the option to hide the prompt when you focus out from the control. The mask prompt is visible when you focus again to the control. The default value of **hidePromptOnLeave** is false.

#### InputMode

The **MaskEdit** supports two type of inputs such as text and password that have been assigned by using the enum values **ej.InputMode.Text** and **ej.InputMode.Password**. The default value for **inputMode** is text in **MaskEdit**.

#### MaskFormat

The **MaskEdit** provides the option to define the **maskFormat** to the value. The default value for **maskFormat** property is empty string.

The following steps explain the implementation of **MaskEdit Properties**.

In the **HTML** page, add a **<div>** element to render the **MaskEdit** widget.

#### HTML

```
<input id="maskEdit" type="text" />
```

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var mask = new ej.MaskEdit($("#maskEdit"), {
    name: "mask",
    inputMode: ej.InputMode.Text,
    maskFormat: "99-999-CCCC",
    customCharacter: "r",
    hidePromptOnLeave: true
  })
}
```

The output for **MaskEdit** with its properties is as follows.



MaskEdit with MaskFormat



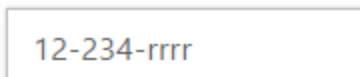
MaskEdit with HidePromptOnLeave



MaskEdit with prompt focus



MaskEdit with InputMode text



MaskEdit with CustomCharacter

### Globalization Support in MaskEdit

We have provided the globalization support in **MaskEdit** control. Our **MaskEdit** control mainly rendered based on the **maskFormat** property, so we have provided the globalization support based on the maskFormat literals. We have given this globalization support option on below maskFormat literals in

**MaskEdit** control. You can change the globalization by using the [locale](#) property. The default value for **locale** property is 'en-US' in **MaskEdit** control.

| Formats | Description   |
|---------|---|
| \$      | Currency symbol value will be changed based on the corresponding culture.   |
| .       | Decimal Separator value will be changed based on the corresponding culture. |
| ,       | Thousand Separator will be changed based on the corresponding culture.      |

To know more about EJ globalize support, please refer the below link

<https://help.syncfusion.com/js/localization>

The following example describes the way to use localization for **MaskEdit** widgets.

Refer the below German culture file in head section of HTML page after the reference of **ej.web.all.min.js** file.

#### JAVASCRIPT

```
<script src="http://cdn.syncfusion.com/{ site.releaseversion
  }/js/i18n/ej.culture.de-DE.min.js"></script>
```

#### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="mask">Mask Edit</label>
</td>
<td>
<input id="maskEdit" type="text" />
</td>
</tr>
</tbody>
</table>
```

#### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var mask = new ej.MaskEdit($("#maskEdit"), {
    locale: "de-De", //specifies the culture for MaskEdit control
    width: "100%",
    maskFormat: "$99,999.99",
    value: "1234567"
  })
}
```

The output for **MaskEdit** with Globalization.

Mask Edit

€12.345,67

MaskEdit with de-DE locale

Mask Edit

\$12,345.67

MaskEdit with en-US locale

## Appearance

### Theme

The **MaskEdit** control's style and appearance can be controlled based on CSS classes. In order to apply styles to the **Textbox** control, you need to refer 2 files namely, **ej.widgets.core.min.css** and **ej.theme.min.css**. If the file **ej.widgets.all.min.css** is referred, then it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.widgets.all.min.css** is the combination of these two.

By default, there are 12 themes support available for **MaskEdit** control namely:

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

### CSS Class

The **CSS** can be customized by using the **cssClass** in the **MaskEdit**. You can customize the **MaskEdit** with **cssClass** property to appear like your desired control.

The following steps explain the implementation of MaskEdit with **cssClass** property.

In the HTML page, add a <div> element to render the MaskEdit widget.

### HTML

```
<input id="maskEdit" type="text" />
```

### JAVASCRIPT

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var mask = new ej.MaskEdit($("#maskEdit"), {
    name: "mask",
    inputMode: ej.InputMode.Text,
    maskFormat: "99-999-99999",
    cssClass: "customCss"
  })
}

```

Customize the CSS properties in custom CSS class.

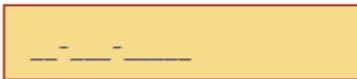
### CSS

```

<style>
.customCss .e-box {
border-color: #9d241b;
}
.customCss .e-input {
background-color: #f6db8d;
}
.customCss .e-select {
background-color: #ecf6ac;
border-color: #3c36e7;
}
</style>

```

The output for MaskEdit after applying **cssClass** is as follows.



### Rounded Corner Support

MaskEdit provides you with rounded corner support whose appearance is different from normal textbox controls.

The following steps explain the implementation of MaskEdit with **showRoundedCorner** property.

In the HTML page, add a <div> element to render the MaskEdit widget.

### HTML

```

<input id="maskEdit" type="text" />

```

### JAVASCRIPT

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var mask = new ej.MaskEdit($("#maskEdit"), {
    name: "mask",
    inputMode: ej.InputMode.Text,
    maskFormat: "99-999-99999",

```



```
showRoundedCorner: true,  
value: "123456"  
})  
}
```

Output of MaskEdit when **showRoundedCorner** is “true”.



### WatermarkText Support

The **MaskEdit** control provide water mark text support. You can display the initial value in the control by water mark.

The following steps explain the implementation of MaskEdit with **watermarkText** property.

In the HTML page, add a <div> element to render the MaskEdit widget.

#### HTML

```
<input id="maskEdit" type="text" />
```

#### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />  
/// <reference path="ej.web.all.d.ts" />  
module EditorComponent {  
  var mask = new ej.MaskEdit($("#maskEdit"), {  
    name: "mask",  
    inputMode: ej.InputMode.Text,  
    maskFormat: "99-999-99999",  
    watermarkText: "Enter the Mask"  
  })  
}
```

Output of MaskEdit when **waterMarkText** is “true”.



### Text Alignment Support

The **MaskEdit** provides text alignment support that allows you to set the alignment of text in the control by using the **textAlign** property.

The following steps explain the implementation of MaskEdit with **textAlign** property.

In the HTML page, add a <div> element to render the MaskEdit widget

#### HTML

```
<input id="maskEdit" type="text" />
```

#### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var mask = new ej.MaskEdit($("#maskEdit"), {
    name: "mask",
    inputMode: ej.InputMode.Text,
    maskFormat: "99-999-99999",
    textAlign: ej.TextAlign.Right
  })
}
```

The output for Textboxes when **textAlign** is set to “right”.



## Menu

### Overview

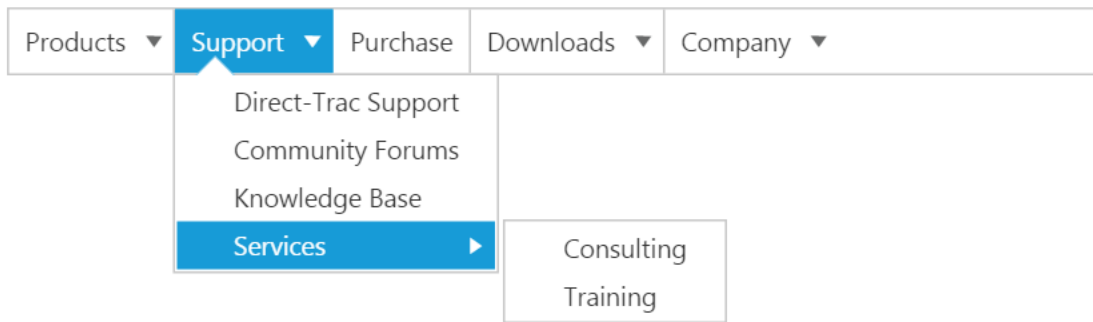
The **Menu** control supports displaying a **Menu** created from list items. The **Menu** is based on a hierarchy of **UL** and **LI** elements where the list items are rendered as sub-menu items.

### Key Features

- **UL/LI template:** Requires only **UL** and **LI** list of items as input.
- **Orientation:** Vertical and horizontal orientation support.
- **Context menu:** A **Menu** that is displayed wherever you right-click on the page or on the control.
- **Keyboard shortcuts:** Allows easy access to **Menu** items through shortcut keys.
- **Data binding:** Supports data binding **Menu** items as remote or local data.
- **Theme:** Essential JavaScript controls feature 12 built-in themes, six flat and six with gradient effects, and also supports custom skin options for user-defined themes.
- **Keyboard navigation:** You can interact with the **Menu** control using the keyboard.
- **RTL support:** This feature allows text in **Menu** items to be displayed from right to left.
- **Center Menu:** This feature allows you to center-align root **Menu** items.
- Separators and open-on-click support.

### Getting Started

This section explains briefly about how to create a **Menu** control in your application with **Typescript**. The **Essential JavaScript Menu** supports displaying a **Menu** of list-out items. This **Menu** is based on ul-li hierarchy, where the sub-list items are rendered as the sub-menu items. The **Menu** control can also be rendered with local and remote data source. From the following guidelines, you can learn how to customize the **Menu** control for a website. In this case, **Syncfusion's** website **Menu** is discussed. The following screenshot displays the appearance of **Menu**.



### Create a Menu

**Essential JavaScript Menu** widgets are basically provided with built-in features like keyboard navigation, show and hide **Menu** items with animations, and flexible API's. From the following guidelines, you can learn how to render **Menu** control with local data source value.

Create an **HTML** page and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

Adding element for **Menu** rendering.

#### HTML

```
<ul id="syncfusionProducts"></ul>
```

#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MenuComponent {
$(function () {
var sample = new ej.Menu($("#syncfusionProducts"));
});
}
```

Output of the above steps.

### Configure parent Menu items

Every **Menu** has a list of **Menu** items with list of sub level **Menu** items. From the following guidelines, you can learn how to initialize the root level elements of **Menu** control with Local data source value. Initialize the **Menu** with data source value as illustrated in the following code example.

#### HTML

```
<ul id="syncfusionProducts"></ul>
```

#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MenuComponent {
  var data = [
    { id: 1, text: "Products", parentId: null },
    { id: 2, text: "Support", parentId: null },
    { id: 3, text: "Purchase", parentId: null },
    { id: 4, text: "Downloads", parentId: null },
    { id: 5, text: "Company", parentId: null }
  ];
  $(function () {
    var sample = new ej.Menu($("#syncfusionProducts"), {
      fields: { dataSource: data, id: "id", parentId: "parentId", text: "text" }
    });
  });
}
```

The following screenshot displays output.

|          |         |          |           |         |
|----------|---------|----------|-----------|---------|
| Products | Support | Purchase | Downloads | Company |
|----------|---------|----------|-----------|---------|

### Initialize sub-level Menu items

Every **Menu** items have a list of sub level **Menu** items. From the following guidelines, you can learn how to initialize the sub level items of **Menu** control. The **parentId** field is used to map root level **Menu** item to its sub level **Menu** item.

The following code example describes how to initialize first level sub menu items of product **Menu** item.

#### HTML

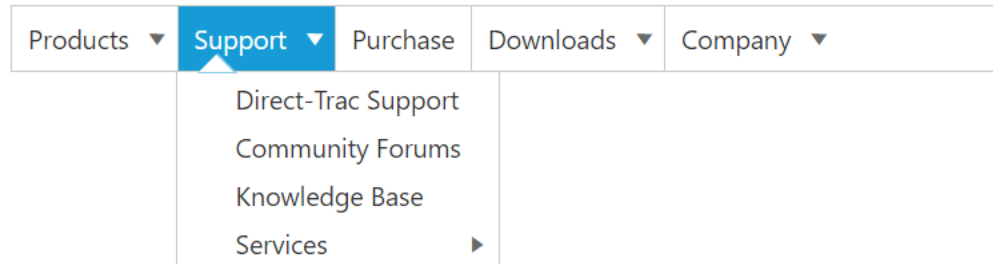
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MenuComponent {
  var data = [
    { id: 1, text: "Products", parentId: null },
    { id: 2, text: "Support", parentId: null },
    { id: 3, text: "Purchase", parentId: null },
    { id: 4, text: "Downloads", parentId: null },
```

```

{ id: 5, text: "Company", parentId: null },
//first level child
{ id: 11, parentId: 1, text: "ASP.NET" },
{ id: 12, parentId: 1, text: "ASP.NET MVC" },
{ id: 13, parentId: 1, text: "Mobile MVC" },
{ id: 14, parentId: 1, text: "Silverlight" },
{ id: 15, parentId: 2, text: "Direct-Trac Support" },
{ id: 16, parentId: 2, text: "Community Forums" },
{ id: 17, parentId: 2, text: "Knowledge Base" },
{ id: 18, parentId: 2, text: "Services" },
{ id: 19, parentId: 4, text: "Evaluation" },
{ id: 20, parentId: 4, text: "Free E-Books" },
{ id: 21, parentId: 4, text: "Metro Studio" },
{ id: 22, parentId: 4, text: "Latest Version" },
{ id: 23, parentId: 5, text: "Technology Resource Portal " },
{ id: 24, parentId: 5, text: "Case Studies" },
{ id: 25, parentId: 5, text: "Bouchers & Datasheets" },
{ id: 26, parentId: 5, text: "FAQ" }
];
$(function () {
var sample = new ej.Menu($("#syncfusionProducts"), {
fields: { dataSource: data, id: "id", parentId: "parentId", text: "text" }
});
});
}

```

Execute the above code example to render the following output.



### Define multiple level Menu items

You can define the sub-menu items to multiple levels in **Menu** control. You need to specify the parent Id value to render sub level **Menu** item for the **Menu** items.

To initialize multiple levels sub menu items, use the following code example.

### HTML

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MenuComponent {
var data = [
{ id: 1, text: "Products", parentId: null },
{ id: 2, text: "Support", parentId: null },
{ id: 3, text: "Purchase", parentId: null },
{ id: 4, text: "Downloads", parentId: null },
{ id: 5, text: "Company", parentId: null },
//first level child

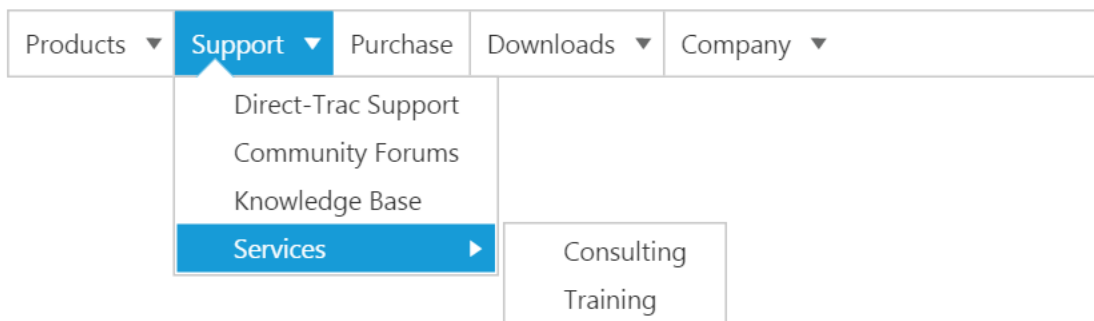
```

```

{ id: 11, parentId: 1, text: "ASP.NET" },
{ id: 12, parentId: 1, text: "ASP.NET MVC" },
{ id: 13, parentId: 1, text: "Mobile MVC" },
{ id: 14, parentId: 1, text: "Silverlight" },
{ id: 15, parentId: 2, text: "Direct-Trac Support" },
{ id: 16, parentId: 2, text: "Community Forums" },
{ id: 17, parentId: 2, text: "Knowledge Base" },
{ id: 18, parentId: 2, text: "Services" },
{ id: 19, parentId: 4, text: "Evaluation" },
{ id: 20, parentId: 4, text: "Free E-Books" },
{ id: 21, parentId: 4, text: "Metro Studio" },
{ id: 22, parentId: 4, text: "Latest Version" },
{ id: 23, parentId: 5, text: "Technology Resource Portal " },
{ id: 24, parentId: 5, text: "Case Studies" },
{ id: 25, parentId: 5, text: "Bouchers & Datasheets" },
{ id: 26, parentId: 5, text: "FAQ" },
//second level child
{ id: 111, parentId: 18, text: "Consulting" },
{ id: 112, parentId: 18, text: "Training" }
];
$(function () {
var sample = new ej.Menu($("#syncfusionProducts"), {
fields: { dataSource: data, id: "id", parentId: "parentId", text: "text" }
});
});
}

```

The following screenshot is the output.



By following the above mentioned steps, you can render the **Menu** control with multiple level sub items through online data source. You can simply customize the **Menu** widget in an efficient manner.

In summary of this getting started, you have now simulated the **Syncfusion's** website **Menu** with **Essential JavaScript Menu**. You have utilized and learn the appearance customization etc.

By following the above mentioned steps, you can render the **Menu** control with multiple level sub items. You can simply customize the **Menu** in an efficient manner.

## Orientation

It gets or sets the direction in which the **Menu** control renders and specifies the orientation of the normal menu. According to the orientation property the **Menu** control renders in horizontal or vertical.

## Horizontal Menu

Horizontal orientation displays the menu items horizontally and it is the default orientation behavior of **Menu** control.

The following steps explain you the details on rendering the **Menu** control.

In an **HTML** page, add the **<UL>** and **<LI>** to configure **Menu** control.

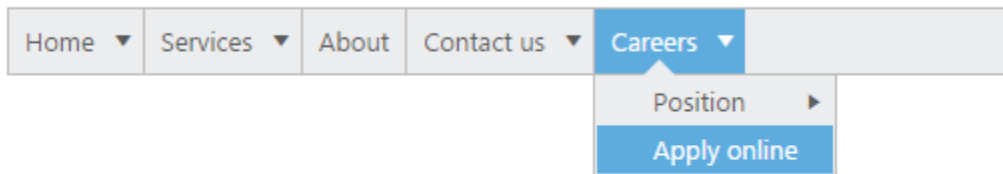
### HTML

```
<div>
<ul id="menu">
<li id="home">
<a href="#">Home</a>
<ul>
<li><a>Foundation</a></li>
<li><a>Launch</a></li>
<li>
<a>About</a>
<ul>
<li><a>Company</a></li>
<li><a>Location</a></li>
</ul>
</li>
</ul>
</li>
<li id="Services">
<a>Services</a>
<ul>
<li><a>Consulting</a></li>
<li><a>Outsourcing</a></li>
</ul>
</li>
<li id="About"><a>About</a></li>
<li id="Contact">
<a>Contact us</a>
<ul>
<li><a>Contact number</a></li>
<li><a>E-mail</a></li>
</ul>
</li>
<li id="Careers">
<a>Careers</a>
<ul>
<li>
<a>Position</a>
<ul>
<li><a>Developer</a></li>
<li><a>Manager</a></li>
</ul>
</li>
<li><a>Apply online</a></li>
</ul>
</li>
</ul>
</div>
```

### JAVASCRIPT

```
// Initialize the control in JavaScript.  
$(function () {  
    var sample = new ej.Menu($("#menu"), {  
        width: 500  
    });  
});
```

The following screenshot displays the output of the above code.



### Vertical Menu

You can also render **Menu** control in vertical direction using **orientation**. To set the vertical orientation of **Menu** control, replace the following script in the above sample code example.

Add the following code in your **<script>** section.

### JAVASCRIPT

```
$(function () {  
    var sample = new ej.Menu($("#menu"), {  
        orientation: ej.Orientation.Vertical  
    });  
});
```

The following screen shot displays the output of the above code.





## Data binding

Data binding enables you to synchronize the elements with different sources of data. You can bind data using two ways, Local data and remote data.

### Field Members

Field is a property that includes the object type. Fields are used to bind the data source and it includes following field members to make binding easier.

List of Field members

| Name           | Description  |
|----------------|--|
| dataSource     | datasource receives Essential DataManager object and JSON object.  |
| Query          | It receives query to retrieve data from the table (query is same as SQL). Example: <code>ej.Query().from("Categories").select("CategoryID,CategoryName").take(3);</code> |
| tableName      | It receives table name to execute query on the corresponding table   |
| Id             | Specifies the id to menu items list  |
| parentId       | Specifies the parent id of the table.  |
| Text           | Specifies the text of menu items list  |
| spriteCssClass | Specifies the sprite CSS class to <code>&lt;li&gt;</code> item list  |
| linkAttribute  | Specifies the link attribute to <code>&lt;a&gt;</code> tag in item list  |
| imageAttribute | Specifies the image attribute to <code>&lt;img&gt;</code> tag inside items list  |
| htmlAttribute  | Specifies the HTML attributes to <code>&lt;li&gt;</code> item list   |
| imageUrl       | Specifies the image URL to <code>&lt;img&gt;</code> tag inside item list.  |

### Local data

To define the local data to the **Menu** control, map the user-defined **JSON** data names with its appropriate **dataSource** column names.

Add the following code in your **HTML** page.

### HTML

```
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<ul id="menu"></ul>
</div>
</div>
</div>
```

### JAVASCRIPT

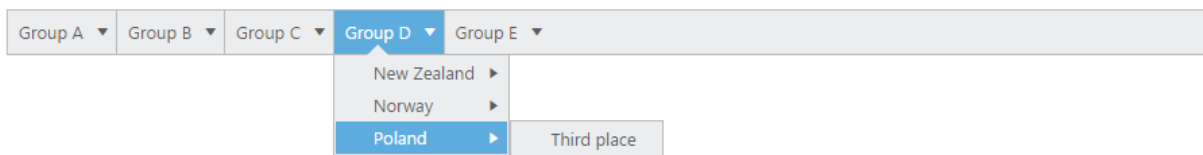
```
// In JavaScript, you can initialize the Menu control.
var data = [
{ id: 1, text: "Group A", parentId: null },
{ id: 2, text: "Group B", parentId: null },
{ id: 3, text: "Group C", parentId: null },
{ id: 4, text: "Group D", parentId: null },
{ id: 5, text: "Group E", parentId: null },
//first level child
{ id: 11, parentId: 1, text: "Algeria" },
{ id: 12, parentId: 1, text: "Armenia" },
{ id: 13, parentId: 1, text: "Bangladesh" },
{ id: 14, parentId: 1, text: "Cuba" },
{ id: 15, parentId: 2, text: "Denmark" },
{ id: 16, parentId: 2, text: "Egypt" },
{ id: 17, parentId: 3, text: "Finland" },
{ id: 18, parentId: 3, text: "India" },
{ id: 19, parentId: 3, text: "Malaysia" },
{ id: 20, parentId: 4, text: "New Zealand" },
{ id: 21, parentId: 4, text: "Norway" },
{ id: 22, parentId: 4, text: "Poland" },
{ id: 23, parentId: 5, text: "Romania" },
{ id: 24, parentId: 5, text: "Singapore" },
{ id: 25, parentId: 5, text: "Thailand" },
{ id: 26, parentId: 5, text: "Ukraine" },
//second level child
{ id: 111, parentId: 11, text: "First Place" },
{ id: 112, parentId: 12, text: "Second Place" },
{ id: 113, parentId: 13, text: "Third place" },
{ id: 114, parentId: 14, text: "Fourth Place" },
{ id: 115, parentId: 15, text: "First Place" },
{ id: 116, parentId: 16, text: "Second Place" },
{ id: 117, parentId: 17, text: "Third Place" },
{ id: 118, parentId: 18, text: "First Place" },
{ id: 119, parentId: 19, text: "Second Place" },
{ id: 120, parentId: 20, text: "First Place" },
{ id: 121, parentId: 21, text: "Second Place" },
{ id: 122, parentId: 22, text: "Third place" },
{ id: 123, parentId: 23, text: "Fourth Place" },
```

```

{ id: 120, parentId: 24, text: "First Place" },
{ id: 121, parentId: 25, text: "Second Place" },
{ id: 122, parentId: 26, text: "Third place" }
];
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module MenuComponent {
$(function () {
var sample = new ej.Menu($("#menu"), {
fields: { dataSource: data, id: "id", parentId: "parentId", text: "text" }
});
});
}

```

The following screenshot displays the output of the above code.



### Remote data

The **Menu** control also provides support for Remote data binding. Here the remote data is placed in Web service and you can render the menu items from the web service using Service **URL**. The data is in **JSON** format.

**DataManager** is used to manage relational data in **JavaScript**. **DataManager** uses two different classes, **ej.DataManager** for processing, and **ej.Query** for serving data. **ej.DataManager** communicates with data source and **ej.Query** generates data queries that are to be read by **DataManager**. To know more about **DataManager** and **Query** refer the following link location.

<https://help.syncfusion.com/js/datamanager>

In the following example, <http://mvc.syncfusion.com/Services/Northwnd.svc/> is used as the **URL**. This acts as web service that is located in the **Syncfusion** server. The web service with the name **Northwind.svc** is used here.

Add the following code in your **HTML** page.

### HTML

```

<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<ul id="shipDetails"></ul>
</div>
</div>

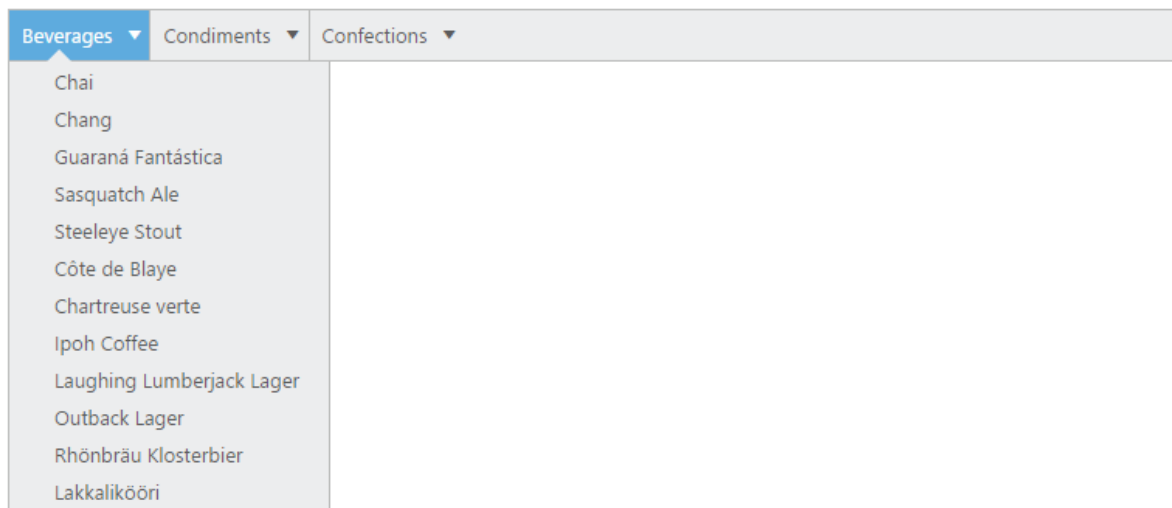
```

```
</div>
```

## JAVASCRIPT

```
// Initialize the Menu control in JavaScript.
$(function () {
    // DataManager creation
    var dataManger = ej.DataManager({
        url: "http://mvc.syncfusion.com/Services/Northwnd.svc/"
    });
    var query =
    ej.Query().from("Categories").select("CategoryID,CategoryName").take(3);
    var sample = new ej.Menu($("#shipDetails"), {
        fields: {
            dataSource: dataManger, query: query, id: "CategoryID", text:
            "CategoryName",
            child: { dataSource: dataManger, tableName: "Products", id: "ProductID",
            parentId: "CategoryID", text: "ProductName" }
        }
    });
});
```

The following screenshot displays the output of the above code.



## Icons and navigation

### Icons

Icons are the images that is displayed in the **Menu** control. To specify the menu with icons you can use **sprite** property to display the icons.

Add the following code in your **HTML** page.

### HTML

```
<div class="content-container-fluid">
<div class="row">
```

```
<div class="cols-sample-area">
<ul id="menu"></ul>
</div>
</div>
</div>
```

## JAVASCRIPT

```
// Initialize the Menu control in JavaScript.
var data = [
{ id: 1, text: "Group A", parentId: null },
{ id: 2, text: "Group B", parentId: null },
{ id: 3, text: "Group C", parentId: null },
{ id: 4, text: "Group D", parentId: null },
{ id: 5, text: "Group E", parentId: null },
//first level child
{ id: 11, parentId: 1, text: "Algeria", sprite: "flag-dz" },
{ id: 12, parentId: 1, text: "Armenia", sprite: "flag-am" },
{ id: 13, parentId: 1, text: "Bangladesh", sprite: "flag-bd" },
{ id: 14, parentId: 1, text: "Cuba", sprite: "flag-cu" },
{ id: 15, parentId: 2, text: "Denmark", sprite: "flag-dk" },
{ id: 16, parentId: 2, text: "Egypt", sprite: "flag-eg" },
{ id: 17, parentId: 3, text: "Finland", sprite: "flag-fi" },
{ id: 18, parentId: 3, text: "India", sprite: "flag-in" },
{ id: 19, parentId: 3, text: "Malaysia", sprite: "flag-my" },
{ id: 20, parentId: 4, text: "New Zealand", sprite: "flag-nz" },
{ id: 21, parentId: 4, text: "Norway", sprite: "flag-no" },
{ id: 22, parentId: 4, text: "Poland", sprite: "flag-pl" },
{ id: 23, parentId: 5, text: "Romania", sprite: "flag-ro" },
{ id: 24, parentId: 5, text: "Singapore", sprite: "flag-sg" },
{ id: 25, parentId: 5, text: "Thailand", sprite: "flag-th" },
{ id: 26, parentId: 5, text: "Ukraine", sprite: "flag-ua" },
];
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MenuComponent {
$(function () {
var sample = new ej.Menu($("#menu"), {
width: 425,
fields: { dataSource: data, id: "id", parentId: "parentId", text: "text",
spriteCssClass: "sprite" }
});
});
}
```

Add the following code in your style section.

## CSS

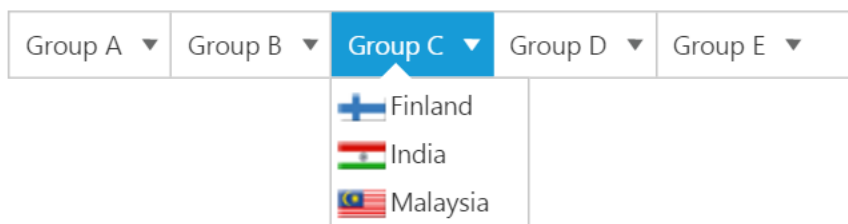
```
<style type="text/css">
#menu {
margin-left: 50px;
}
.e-menu li > ul > li > a {
padding: 3px 24px 3px 35px;
}
```

```

[class^="flag-"],
[class*="flag-"] {
background-image: url("../content/images/autocomplete/flags.png");
height: 14px;
left: 2px;
top: 4px;
width: 24px;
}
.flag-am {background-position: -25px 0;}
.flag-bd {background-position: -75px 0;}
.flag-cu {background-position: -25px -15px;}
.flag-dk {background-position: -50px -15px;}
.flag-dz {background-position: -75px -15px;}
.flag-eg {background-position: -125px -15px;}
.flag-fi {background-position: -25px -30px;}
.flag-id {background-position: -100px -30px;}
.flag-in {background-position: -125px -30px;}
.flag-my {background-position: -25px -45px;}
.flag-no {background-position: -75px -45px;}
.flag-nz {background-position: -100px -45px;}
.flag-pl {background-position: -125px -45px;}
.flag-ro {background-position: -50px -60px;}
.flag-sg {background-position: -100px -60px;}
.flag-th {background-position: -125px -60px;}
.flag-ua {background-position: -25px -75px;}
</style>

```

The following screenshot displays the output for the above code.



**Note:** Images for this sample are available in (installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\samples\web\content\images<br/>

### Navigation

Navigation in Menu control is the default usage to navigate into the other web page. You can navigate to another page in menu item by providing link to the menu items. Navigation in **Menu** control can be achieved by placing “**href**” path to the anchor tag. Use the following code sample for navigating in **Menu** control.

Add the following code in your **HTML** page.

### HTML

```

<div>
<ul id="link">
<li id="searchengine">
<a href="#">Search engine</a>

```

```

<ul>
<li><a href="http://www.bing.com/">Bing</a></li>
<li><a href="https://www.google.co.in/">Google</a></li>
<li><a href="https://in.yahoo.com/">Yahoo</a></li>
<li><a href="http://www.rediff.com/">Rediff</a></li>
</ul>
</li>
<li id="atd"><a href="http://allthingsd.com/">All things digital</a></li>
<li id="electronics">
<a>Electronics</a>
<ul>
<li>
<a href="http://www.engadget.com/">Engadget</a>
</li>
<li><a href="http://www.electronista.com/">Electronista</a></li>
<li><a href="http://www.gearlog.com/">Gearlog</a></li>
</ul>
</li>
<li id="cnet"><a href="http://www.centernetworks.com/">Center
networks</a></li>
<li id="webpronews">
<a href="http://www.webpronews.com/">Webpronews</a>
</li>
</ul>
</div>

```

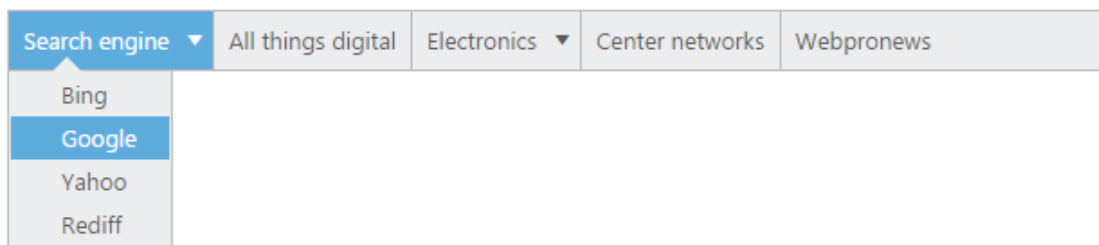
## JAVASCRIPT

```

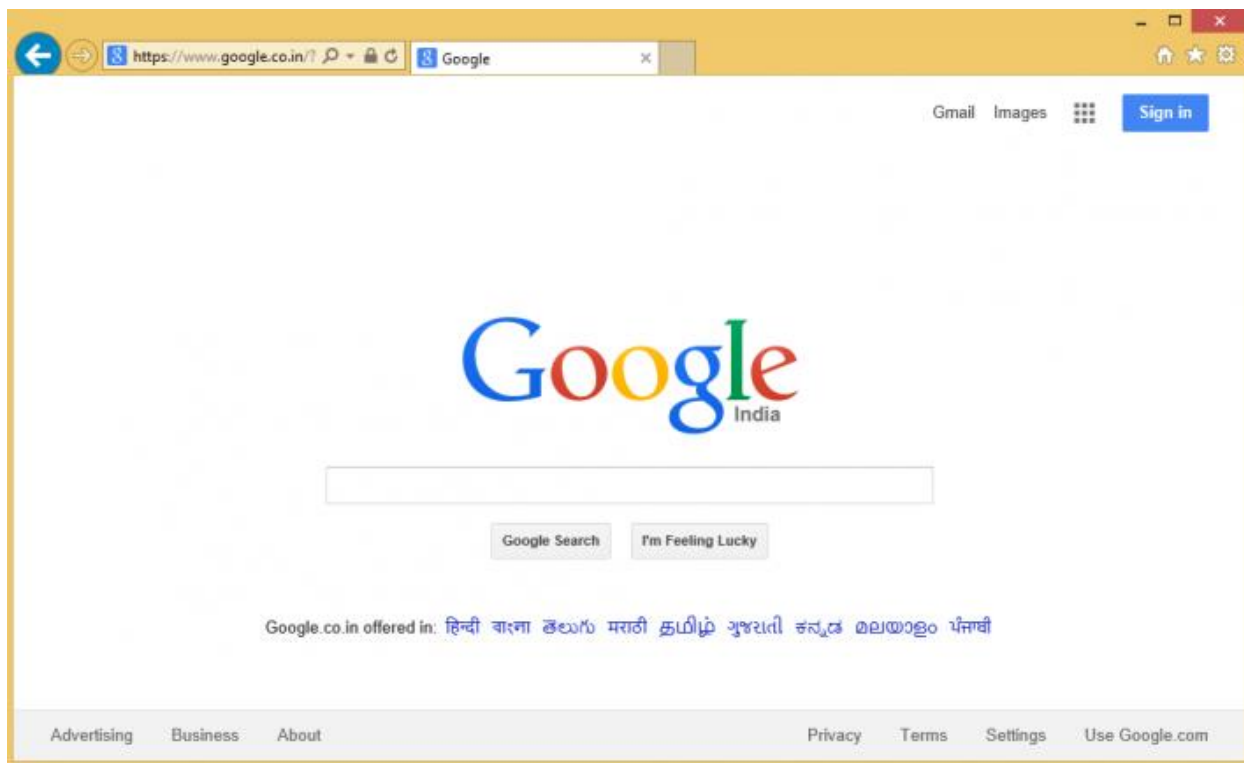
// Add the following code in your script section.
$(function () {
var sample = new ej.Menu($("#link"), {
width: 612 });
});

```

The following screenshot displays the output for the above code example.



When you click on **“Google”** that is present under **“Search engine”**, it navigates to the link that you specified in the sample code. Then the output is as follows.



### Customizing the Submenu direction

You can customize the direction to open the sub menu items using **subMenuDirection** property. **subMenuDirection** accepts the type as string or enum and value as “Left” and “Right”.

In the following example, the Sub menus opens in the Left side of the menu.

Add the following code in your **HTML** page.

#### HTML

```
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<ul id="coProducts">
<li id="Products">
<a href="#">Products</a>
<ul>
<li><a>ASP.NET</a></li>
<li><a>ASP.NET MVC</a></li>
<li><a>WinRT (XMAL)</a></li>
<li><a>Silverlight</a></li>
</ul>
</li>
<li id="Support">
<a>Support</a>
<ul>
<li><a>Direct-Trac Support</a></li>
<li>
<a>Services</a>

```



```
<ul>
<li><a>Consulting</a></li>
<li><a>Training</a></li>
</ul>
</li>
</ul>
</li>
<li id="Purchase"><a>Purchase</a></li>
<li id="Downloads">
<a>Downloads</a>
<ul>
<li><a>Evaluation</a></li>
</ul>
</li>
<li id="Resources">
<a>Resources</a>
<ul>
<li>
<a>Technology Resource Portal </a>
<ul>
<li><a>White Papers</a></li>
</ul>
</li>
<li><a>FAQ</a></li>
</ul>
</li>
<li id="Company">
<a>Company</a>
<ul>
<li>
<a>About Us</a>
<ul>
<li><a>Media Kit</a></li>
</ul>
</li>
<li><a>Company Blog</a></li>
<li><a>Technical Blog</a></li>
<li><a>Newsletter</a></li>
<li>
<a>Partners</a>
<ul>
<li><a>Technology Partners</a></li>
<li><a>Training Partners</a></li>
<li><a>Consulting Partners</a></li>
</ul>
</li>
<li>
<a>Locations</a>
<ul>
<li><a>RDU</a></li>
</ul>
</li>
<li><a>Contact Us</a></li>
<li><a>Careers</a></li>
</ul>
</li>
</ul>
```

```

</div>
</div>
</div>

```

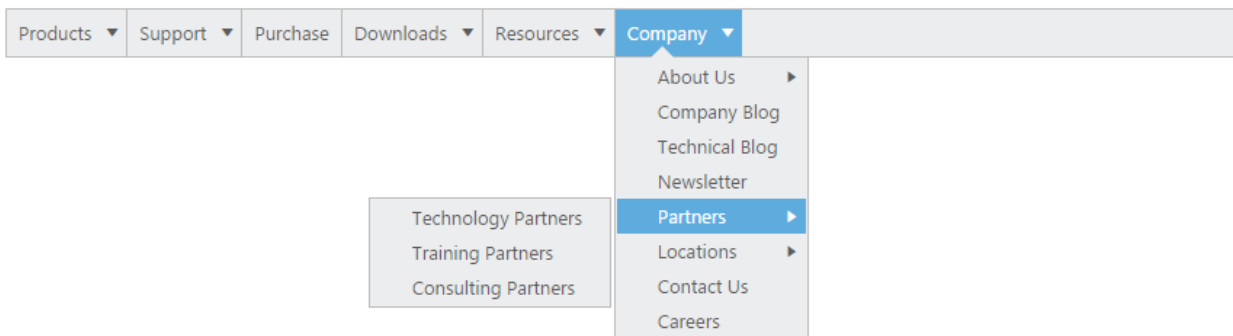
## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
/// <reference path="../tsfiles/jquery.d.ts" />
/// <reference path="../tsfiles/ej.web.all.d.ts" />
module MenuComponent {
$(function () {
var sample = new ej.Menu($("#coProducts"), {
subMenuDirection: "left"
});
});
}

```

The output for the above code example is as follows.



You can even achieve auto positioning for Context Menu. Use the following code sample for context menu in order to open the submenu items of context menu in left side.

Add the following code in your **HTML** page.

## HTML

```

<div>
<div id="target" class="textarea">
HTML is written in the form of HTML elements consisting of tags enclosed in
angle
brackets (like
&#60;html&#62;
), within the web page content. HTML tags most commonly come in pairs like
and , although
some tags, known as empty elements, are unpaired, for example

```

```

    &#60;img&#62;. The purpose of a web browser is to read HTML documents and
    compose them into
    visible or audible web pages. The browser does not display the HTML tags,
    but uses
    the tags to interpret the content of the page.
</div>
<ul id="contextMenu">
<li>
<a>Cut</a>
<ul>
<li>
<a>Cut Particular</a>
</li>
<li><a>Cut Fully</a></li>
</ul>
</li>
<li><a>Copy</a></li>
<li><a>Paste</a></li>
<li class="separator"></li>
<li><a>Comments</a></li>
<li><a>Links</a></li>
<li><a>Clear Formatting</a></li>
</ul>
</div>

```

### JAVASCRIPT

```

// Add the following code in your script section.
$(function () {
var sample = new ej.Menu($("#contextMenu"), {
menuType: ej.MenuType.ContextMenu,
openOnClick: false,
contextMenuTarget: "#target",
subMenuDirection: ej.Direction.Left
});
});

```

Add the following code in your style section.

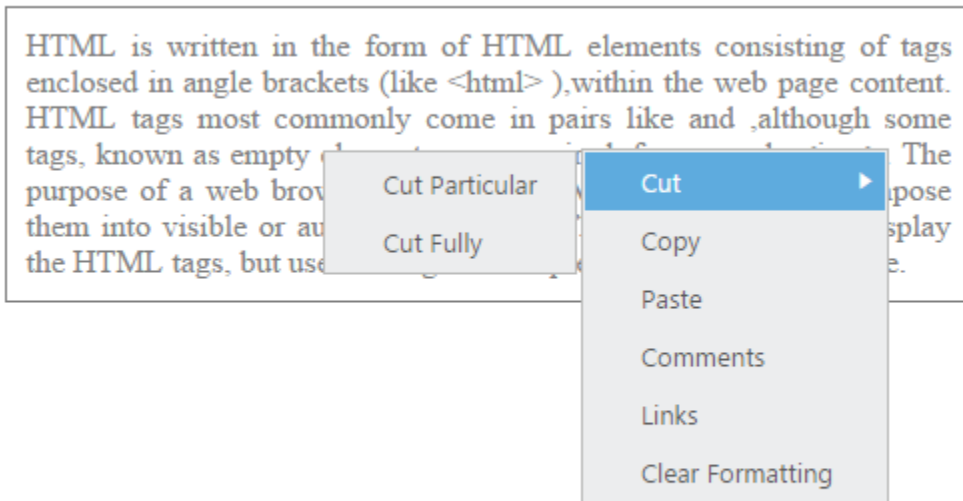
### CSS

```

<style type="text/css">
.textarea {
border: 1px solid;
padding: 10px;
position: relative;
text-align: justify;
width: 463px;
color: gray;
margin: 0 auto;
}
</style>

```

The output for the above code example is as follows.



## Look and feel

**Essential JavaScript** controls feature 12 built-in themes, six flat and gradient effects, and also supports custom skin options for user-defined themes.

12 themes support available for **Menu** control namely, *default-theme* *flat-azure-dark* *flat-lime* *flat-lime-dark* *flat-saffron* *flat-saffron-dark* *gradient-azure* *gradient-azure-dark* *gradient-lime* *gradient-lime-dark* *gradient-saffron* *gradient-saffron-dark*

## cssClass

**Menu** control also customizes its appearance using user-defined **CSS** and custom skin options (colors and backgrounds). To apply custom themes you can use the “cssClass” property. “cssClass” property sets the root class for **Menu** control theme.

Using this **cssClass** you can override the existing styles under the theme style sheet. The theme stylesheet applies theme-specific styles like colors and backgrounds. In the following sample the value of “cssClass” property is set as “Purple-dark”. Purple-dark is added as root class to **Menu** control at the runtime. From this root class you can customize the **Menu** control theme.

Add the following code in your **HTML** page.

## HTML

```
<div>
<div>
<ul id="menu">
<li id="home">
<a href="#">Home</a>
<ul>
<li><a>Foundation</a></li>
<li><a>Launch</a></li>
<li>
```

```

<a>About</a>
<ul>
<li><a>Location</a></li>
</ul>
</li>
</ul>
</li>
<li id="Services">
<a>Services</a>
<ul>
<li><a>Outsourcing</a></li>
</ul>
</li>
<li id="About"><a>About</a></li>
<li id="Contact">
<a>Contact us</a>
<ul>
<li><a>E-mail</a></li>
</ul>
</li>
<li id="Careers">
<a>Careers</a>
<ul>
<li>
<a>Position</a>
<ul>
<li><a>Developer</a></li>
</ul>
</li>
<li><a>Apply online</a></li>
</ul>
</li>
</ul>
</div>
</div>

```

## JAVASCRIPT

```

// Add the following code in your script section.
$(function () {
var sample = new ej.Menu($("#menu"), {
width: 500,
cssClass: "Purple-dark"
});
});

```

Add the following code in your style section.

## CSS

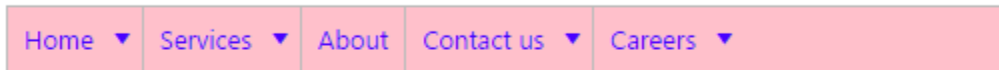
```

<style type="text/css" class="cssStyles">
.Purple-dark .e-menu,.e-menu.e-horizontal .e-list > ul {
background: pink;
}
.Purple-dark .e-menu.e-horizontal .e-list > a {
color: blue;
}

```

```
}
</style>
```

Following screenshot displays the output of the above code.



## Background Template

**Menu** control also provides support for template support. Normally **Menu** control can be created by using **UL** and **LI** tags in the preferred way. But in template supporting, you can customize the appearance of sub menu items rendering.

Initialize the Template **Menu** as illustrated in the following code example.

Add the following code in your **HTML** page.

### HTML

```
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<ul id="template">
<li>
<a>My Computer</a>
<ul>
<li>
<div class="temp temp1">
<span>Disks</span>
<ul>
<li><a>Local Disk : C</a></li>
<li><a>Local Disk : D</a></li>
</ul>
</li>
<li><a>Local Disk : E</a></li>
<li><a>Local Disk : F</a></li>
</ul>
</div>
</li>
</ul>
</li>
<li>
<a>Libraries</a>
<ul>
<li>
<div class=" temp temp2">
<div>
```

```

<span>Documents</span>
<ul>
<li><a>Images</a></li>
<li><a>Videos</a></li>
</ul>
<ul>
<li><a>Documents</a></li>
<li><a>Music</a></li>
</ul>
</div>
</div>
</li>
</ul>
</li>
<li>
<a>Favorites </a>
<ul>
<li>
<div class="temp temp3">
<div>
<span>Favorites</span>
<ul>
<li><a>Downloads</a></li>
<li><a>Recent Places</a></li>
</ul>
<ul>
<li><a>Desktop</a></li>
</ul>
</div>
</div>
</li>
</ul>
</li>
<li>
<a>Gaming</a>
<ul>
<li>
<div class="temp temp4">
<div>
<span>GAMING</span>
<ul>
<li><a>Upcoming</a></li>
<li><a>Consoles</a></li>
</ul>
<ul>
<li><a>FIFA 2999</a></li>
<li><a>Carom legend</a></li>
</ul>
</div>
</div>
</li>
</ul>
</li>
</ul>
</div>
</div>
</div>

```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MenuComponent {
  $(function () {
    var sample = new ej.Menu($("#template"), {
    });
  });
}
```

Add the following code in your style section.

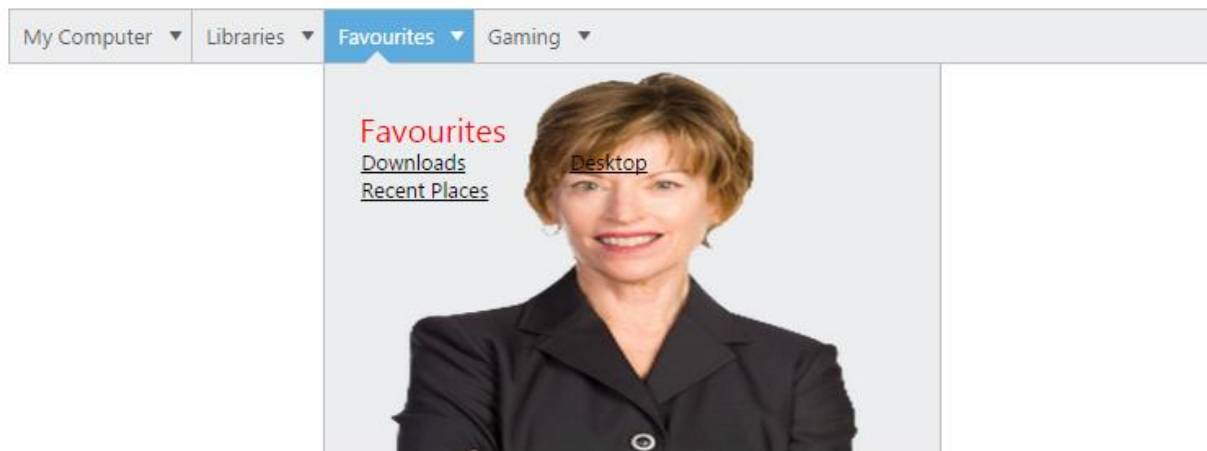
**CSS**

```
<style type="text/css">
.temp {
height: 237px;
width: 375px;
font-family: segoe UI;
cursor: default;
background-size: 100% 100%;
}
.temp span {
color: red;
float: left;
font-size: 20px;
left: 20px;
position: relative;
top: 25px;
width: 100px;
}
.temp ul {
float: left;
font-size: 14px;
left: -79px;
list-style-type: none;
margin: 0;
padding: 0;
position: relative;
top: 50px;
width: 128px;
}
.temp ul li {
font-size: 13px;
}
.temp ul li a {
text-decoration: underline;
cursor: pointer;
color: #000;
}
.temp1 {
background-image: url("1.jpg");
}
.temp2 {
```



```
background-image: url("2.jpg");
}
.temp3 {
background-image: url("3.jpg");
}
.temp4 {
background-image: url("4.jpg");
}
.e-menu.e-horizontal li > ul, .e-menu.e-horizontal li > ul > li:hover {
background-color: #fff;
}
.e-menu.e-horizontal > li > ul:after {
border-color: transparent transparent #fff;
}
</style>
```

Execute the above code to render the following output.



## Context Menu

A context menu is a type of menu in a graphical user interface (GUI) that appears when you perform right click operation. In this **Menu** control you can use a context menu by specifying the type of menu as **ContextMenu**. A context also provides support for nested level of menu items.

Before you use the context menu, provide the target area for it.

In the following example, a context menu for the division containing text is created. In this, when you perform right click operation, the following menu appears with open, edit, etc.

Add the following code in your **HTML** page.

### HTML

```
<div>
<div id="target" class="textarea">
HTML is written in the form of HTML elements consisting of tags enclosed in
angle
brackets (like
```

```

<#60;html#62;
),within the web page content. HTML tags most commonly come in pairs like
and ,although
some tags, known as empty elements, are unpaired, for example
<#60;img#62;. The purpose of a web browser is to read HTML documents and
compose them into
visible or audible web pages. The browser does not display the HTML tags,
but uses
the tags to interpret the content of the page.
</div>
<ul id="contextMenu">
<li><a>Open</a></li>
<li><a>Edit</a></li>
<li><a>Save</a></li>
<li class="separator"></li>
<li><a>Save as</a></li>
<li><a>Print</a></li>
<li><a>Properties</a></li>
</ul>
</div>

```

## JAVASCRIPT

```

// Add the following code in your script section.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MenuComponent {
$(function () {
var sample = new ej.Menu($("#contextMenu"), {
menuType: ej.MenuType.ContextMenu,
openOnClick: false,
contextMenuTarget: "#target"
});
});
}

```

Add the following code in your style section.

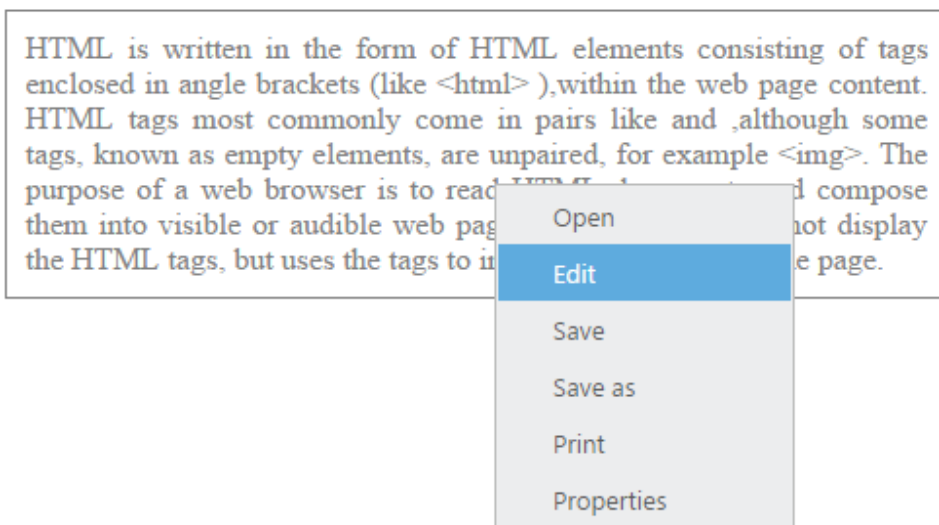
## CSS

```

<style type="text/css">
.textarea {
border: 1px solid;
padding: 10px;
position: relative;
text-align: justify;
width: 463px;
color: gray;
margin: 0 auto;
}
</style>

```

The following screen shot displays the output of the above code.



You can hide and show the context menu using the following methods.

#### HideContextMenu

Hides the context menu control. Add the following script code in the sample in order to hide the context menu.

#### JAVASCRIPT

```
$(function () {  
    var sample = new ej.Menu($("#contextMenu"), {  
        menuType: ej.MenuType.ContextMenu,  
        openOnClick: false,  
        contextMenuTarget: "#target"  
    });  
    sample.hide();  
});
```

#### ShowContextMenu

Shows the context menu control. Add the following script code in the sample in order to show the context menu.

#### JAVASCRIPT

```
$(function () {  
    var sample = new ej.Menu($("#contextMenu"), {  
        menuType: ej.MenuType.ContextMenu,  
        openOnClick: false,  
        contextMenuTarget: "#target"  
    });  
    sample.show();  
});
```

## Center Menu

You can align the **Menu** items to center by setting “enableCenterAlign” property as true.

“**enableCenterAlign**” property accepts Boolean value. By default, its value is **false**. When set to **true**, then the root menu items is aligned in center.

Add the following code in your **<script>** section.

### HTML

```
<div>
<ul id="menu">
<li id="home">
<a href="#">Home</a>
<ul>
<li><a>Foundation</a></li>
<li><a>Launch</a></li>
<li>
<a>About</a>
<ul>
<li><a>Company</a></li>
<li><a>Location</a></li>
</ul>
</li>
</ul>
</li>
<li id="Services">
<a>Services</a>
<ul>
<li><a>Consulting</a></li>
<li><a>Outsourcing</a></li>
</ul>
</li>
<li id="About"><a>About</a></li>
<li id="Contact">
<a>Contact us</a>
<ul>
<li><a>Contact number</a></li>
<li><a>E-mail</a></li>
</ul>
</li>
<li id="Careers">
<a>Careers</a>
<ul>
<li>
<a>Position</a>
<ul>
<li><a>Developer</a></li>
<li><a>Manager</a></li>
</ul>
</li>
<li><a>Apply online</a></li>
</ul>
</li>
</ul>
</div>
```

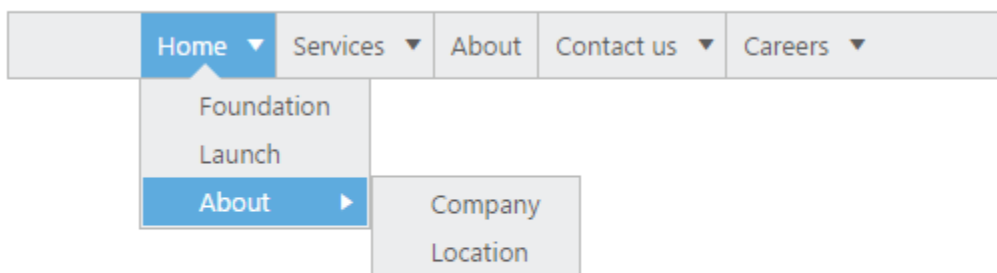
**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module MenuComponent {
$(function () {
var sample = new ej.Menu($("#menu"), {
width: 500,
enableCenterAlign: true
});
});
}

```

The following screenshot displays the output of the above code.



## RTL Support

The **enableRTL** option allows the **Menu** control to display it in the right to left direction. By default, this option is set to “false” in the **Menu** control.

The following code depicts you on how to enable the **rtl** property of the **Menu** control.

**HTML**

```

<div>
<ul id="menu">
<li id="home">
<a href="#">Home</a>
<ul>
<li><a>Foundation</a></li>
<li><a>Launch</a></li>
<li>
<a>About</a>
<ul>
<li><a>Company</a></li>
<li><a>Location</a></li>
</ul>
</li>
</ul>
</li>

```

```

<li id="Services">
  <a>Services</a>
  <ul>
    <li><a>Consulting</a></li>
    <li><a>Outsourcing</a></li>
  </ul>
</li>
<li id="About"><a>About</a></li>
<li id="Contact">
  <a>Contact us</a>
  <ul>
    <li><a>Contact number</a></li>
    <li><a>E-mail</a></li>
  </ul>
</li>
<li id="Careers">
  <a>Careers</a>
  <ul>
    <li>
      <a>Position</a>
      <ul>
        <li><a>Developer</a></li>
        <li><a>Manager</a></li>
      </ul>
    </li>
    <li><a>Apply online</a></li>
  </ul>
</li>
</ul>
</div>

```

## JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module MenuComponent {
  $(function () {
    var sample = new ej.Menu($("#menu"), {
      enableRTL: true
    });
  });
}

```

Following screenshot displays the output for the above code.



RTL Support

## Separators

Menu can also contain separators that are horizontal bars between menu items. You cannot select a separator. Separators are somewhat similar to [borders](#), except that they are genuine components and, as such, are drawn inside a control, rather than around the edges of the **Menu** control. **enableSeparator** is the property that is used to display the separators in the **Menu** control. It accepts the Boolean type value. Its default value is true.

Add the following **<script>** in the above code sample to display the **Menu** control without separator by setting **enableSeparator** as **false**.

### HTML

```
<div>
<ul id="menu">
<li id="home">
<a href="#">Home</a>
<ul>
<li><a>Foundation</a></li>
<li><a>Launch</a></li>
<li>
<a>About</a>
<ul>
<li><a>Company</a></li>
<li><a>Location</a></li>
</ul>
</li>
</ul>
</li>
<li id="Services">
<a>Services</a>
<ul>
<li><a>Consulting</a></li>
<li><a>Outsourcing</a></li>
</ul>
</li>
<li id="About"><a>About</a></li>
<li id="Contact">
<a>Contact us</a>
<ul>
<li><a>Contact number</a></li>
<li><a>E-mail</a></li>
</ul>
</li>
<li id="Careers">
<a>Careers</a>
<ul>
<li>
<a>Position</a>
<ul>
<li><a>Developer</a></li>
<li><a>Manager</a></li>
</ul>
</li>
<li><a>Apply online</a></li>
</ul>
</li>
</ul>
</div>
```

### JAVASCRIPT

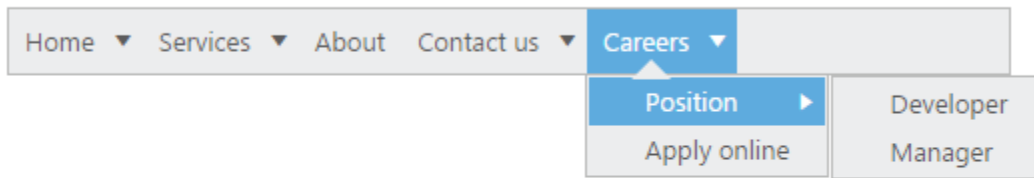
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```

module MenuComponent {
  $(function () {
    var sample = new ej.Menu($("#menu"), {
      width: 500,
      enableSeparator: false
    });
  });
}

```

The following screenshot displays the output for the above code.



## Separators for Context Menu

We can add the separators for particular ContextMenu items by including **e-separator** class in the required LI elements. Add the following code to display ContextMenu with separator lines.

### HTML

```

<div id="target" class="textarea">
HTML is written in the form of HTML elements consisting of tags enclosed in
angle brackets (like <html>), within the web page content. HTML tags
most commonly come in pairs like and , although some tags, known as empty
elements, are unpaired, for example <img>. The purpose of a web
browser is to read HTML documents and compose them into visible or audible
web pages. The browser does not display the HTML tags, but uses the tags to
interpret the content of the page.
</div>
<ul id="contextMenu">
<li><a>Cut</a></li>
<li><a>Copy</a></li>
<li class="e-separator"><a>Paste</a></li>
<li><a>Comments</a></li>
<li><a>Links</a></li>
<li><a>Clear Formatting</a></li>
</ul>

```

### JAVASCRIPT

```

<script type="text/javascript">
$(function () {
  var sample = new ej.Menu($("#contextMenu"), {

```



```

menuType: ej.MenuType.ContextMenu,
openOnClick: false,
contextMenuTarget: "#target",
});
});
</script>

```

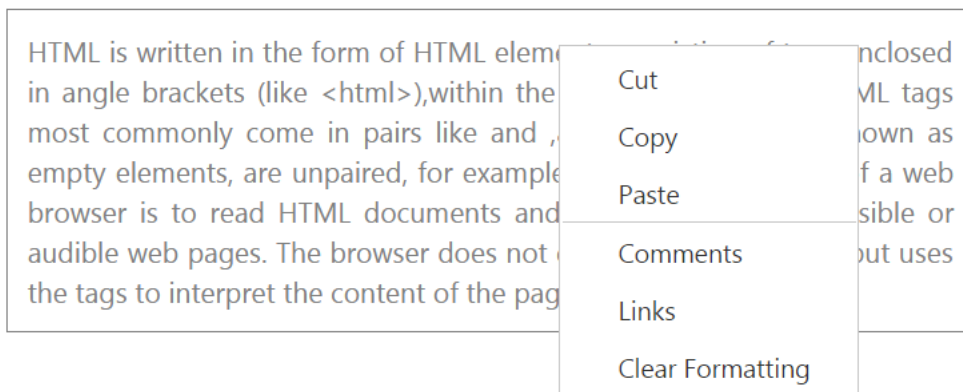
## CSS

```

<style type="text/css">
.textarea {
border: 1px solid;
padding: 10px;
position: relative;
text-align: justify;
width: 463px;
color: gray;
}
</style>

```

The following screenshot displays the output for the above code.



## Responsive Layout

Responsive Layout is aimed at crafting sites to provide an optimal viewing experience—easy reading and navigation with a minimum of resizing, panning, and scrolling—across a wide range of devices (from mobile phones to desktop computer monitors). In order to get responsive layout, you can add **ej.responsive.css** file in this sample. **CDN** link for the responsive CSS file is as follows.

```

[http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/responsive-
css/ej.responsive.css](http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/responsive-
css/ej.responsive.css)

```

---

**Note:** Refer to the ej.responsive.css file after the ej.widgets.all.min.css file

---

Add the above **css** link in the code sample.

Add the following code in your **HTML** page.

## HTML

```

<div>
<ul id="menu">
<li id="home">
<a href="#">Home</a>
<ul>
<li><a>Foundation</a></li>
<li><a>Launch</a></li>
<li>
<a>About</a>
<ul>
<li><a>Company</a></li>
<li><a>Location</a></li>
</ul>
</li>
</ul>
</li>
<li id="Services">
<a>Services</a>
<ul>
<li><a>Consulting</a></li>
<li><a>Outsourcing</a></li>
</ul>
</li>
<li id="About"><a>About</a></li>
<li id="Contact">
<a>Contact us</a>
<ul>
<li><a>Contact number</a></li>
<li><a>E-mail</a></li>
</ul>
</li>
<li id="Careers">
<a>Careers</a>
<ul>
<li>
<a>Position</a>
<ul>
<li><a>Developer</a></li>
<li><a>Manager</a></li>
</ul>
</li>
<li><a>Apply online</a></li>
</ul>
</li>
</ul>
</div>

```

## JAVASCRIPT

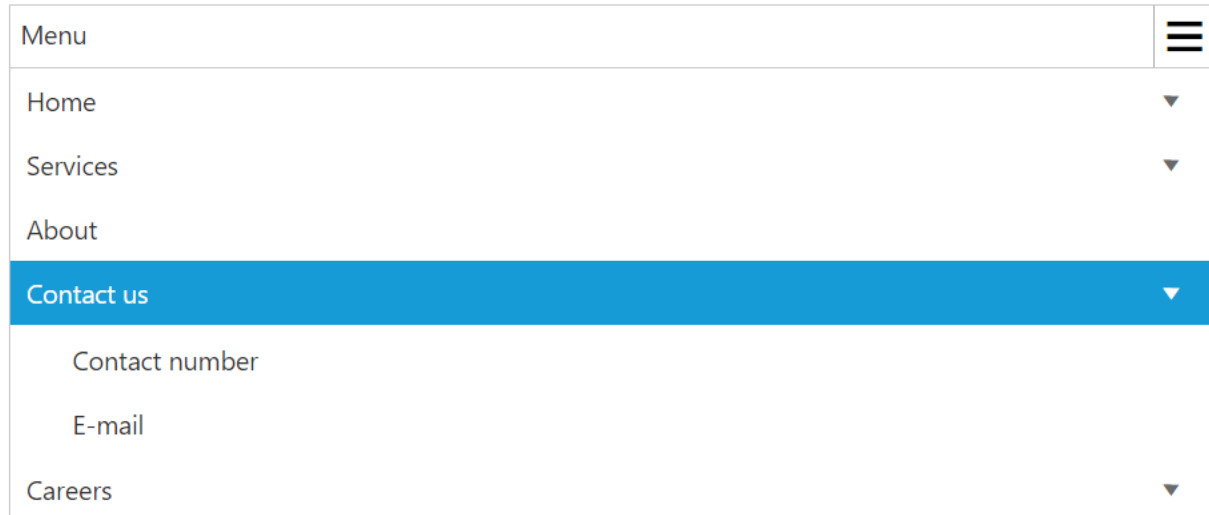
```

// Add the following code in your script section.
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module MenuComponent {
$(function () {
var sample = new ej.Menu($("#menu"), {
});

```

```
});
}
```

The following screenshot displays the output on executing the above code



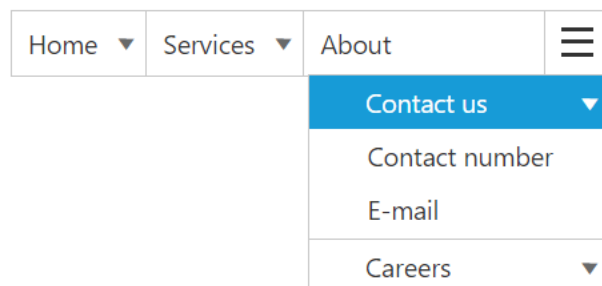
#### Responsive in Desktop:

When menu width is small and window width is normal as desktop ,only the overflown menu items will be moved inside menu popup.You can also set width and height for popup menu using **overflowHeight** and **overflowWidth** API

#### JAVASCRIPT

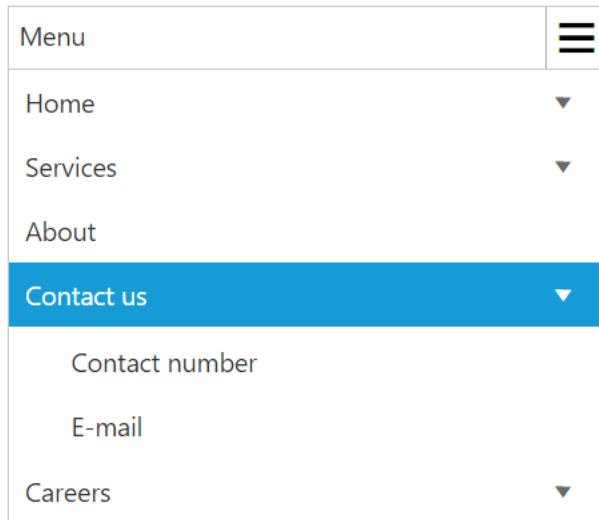
```
// Add the following code in your script section.
$(function () {
  var sample = new ej.Menu($("#menu"), {
    width: "300px"
  });
});
```

The following output shows the output of the above code



#### Responsive in Mobile or Tablet:

Menu will be displayed in mobile or Tablet as shown in the below image:



**Note:** Window width below 767px is considered as Mobile or Tablet mode in our menu.

### Keyboard Navigation

The **Menu** control also provides support for keyboard navigation. In the **Menu** control, it is possible to control the entire menu control items by using the provided shortcut keys.

The various keyboard shortcuts available within the **Menu** control are discussed in the following table, List of keyboard shortcut keys

| Keys                          | Usage                           |
|-------------------------------|---------------------------------|
| Esc                           | Closes the opened Menu control. |
| Enter                         | Selects the focused item.       |
| Up/left/down/right arrow keys | Navigates up or previous item.  |
| Down                          | Navigates down or next item.    |
| Left                          | Navigates to previous group.    |
| Right                         | Navigates to next group.        |

Add the following code for Keyboard navigation in your **Menu** control.

### HTML

```
<div>
<div>
<ul id="menu">
<li id="home">
<a href="#">Home</a>
<ul>
<li><a>Foundation</a></li>
<li><a>Launch</a></li>
<li>
<a>About</a>
```

```
<ul>
<li><a>Company</a></li>
<li><a>Location</a></li>
</ul>
</li>
</ul>
</li>
<li id="Services">
<a>Services</a>
<ul>
<li><a>Consulting</a></li>
<li><a>Outsourcing</a></li>
</ul>
</li>
<li id="About"><a>About</a></li>
<li id="Contact">
<a>Contact us</a>
<ul>
<li><a>Contact number</a></li>
<li><a>E-mail</a></li>
</ul>
</li>
<li id="Careers">
<a>Careers</a>
<ul>
<li>
<a>Position</a>
<ul>
<li><a>Developer</a></li>
<li><a>Manager</a></li>
</ul>
</li>
<li><a>Apply online</a></li>
</ul>
</li>
</ul>
</div>
</div>
```

## JAVASCRIPT

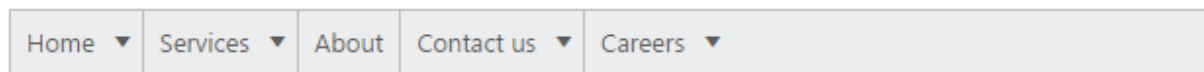
```
// Add the following code in your script section.
$(function () {
var sample = new ej.Menu($("#menu"), {
//Control focus key
$(document).on("keydown", function(e) {
if (e.altKey && e.keyCode === 74) { // j- key code.
$("#menu").focus();
}
});
});
```

Add the following code in your style section.

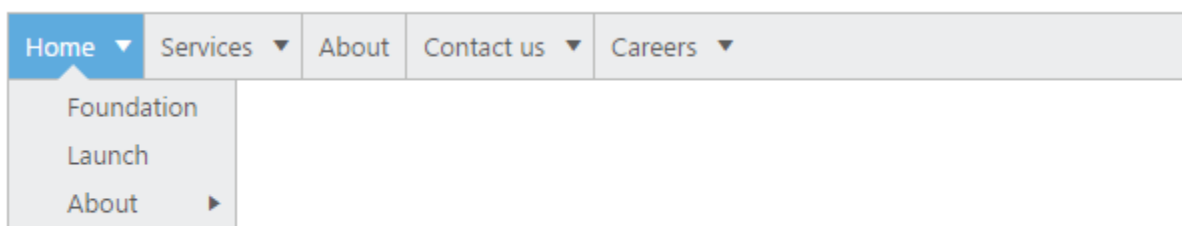
## CSS

```
<style type="text/css">
#keyboard {
margin-left: 50px;
}
</style>
```

Following screenshot displays the output of the above code.



When you press **alt+j**, the first item of the **Menu** control only gets focused as displayed in the following screenshot.



Similarly you can access the **Menu** control using keyboard itself.

## Miscellaneous

### Height

Specifies the height of the root menu. You can customize the height of the **Menu** control by using height property.

You can specify the height of the **Menu** control in the script as follows.

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module MenuComponent {
$(function () {
var sample = new ej.Menu($("#menu"), {
height: 50
});
});
}
```

### Width

Specifies the width of the main menu. You can customize the width of the **Menu** control by using width property.

You can specify the width of the **Menu** control in the script as follows.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.Menu($("#menu"), {  
    width: 700  
  });  
});
```

### Open on click

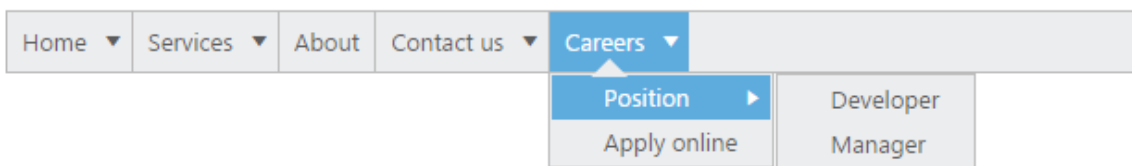
Specifies the sub menu items to be show or open only on click. It accepts the Boolean value. Its default value is false. If we set “**openOnClick**” property to true then the submenu items will open only on click. By default the submenu will open when we hover on menu items.

Add the following **<script>** in the above code sample.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.Menu($("#menu"), {  
    width: 612,  
    openOnClick: true  
  });  
});
```

Output screenshot for the above code example is as follows.



### Animation

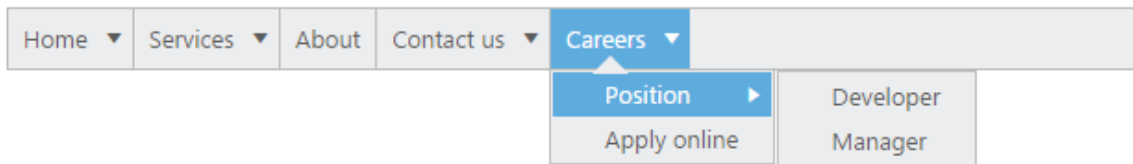
**AnimationType** is used to enable or disable the Animation when hover or click on menu items. Its value type is string. It accepts two values such as “none” and “default”. Support to disable the **AnimationType** when hover or click on menu items is none. Support to enable the Animation Type when hover or click on menu items is default.

Add the following **<script>** in the above sample code.

## JAVASCRIPT

```
$(function () {  
  var sample = new ej.Menu($("#menu"), {  
    width: 612,  
    animationType: ej.AnimationType.Default  
  });  
});
```

Output screenshot for the above code sample is as follows.



### Title text

Specifies the title to the responsive menu. You can provide title to the **Menu** control by using **titleText** property.

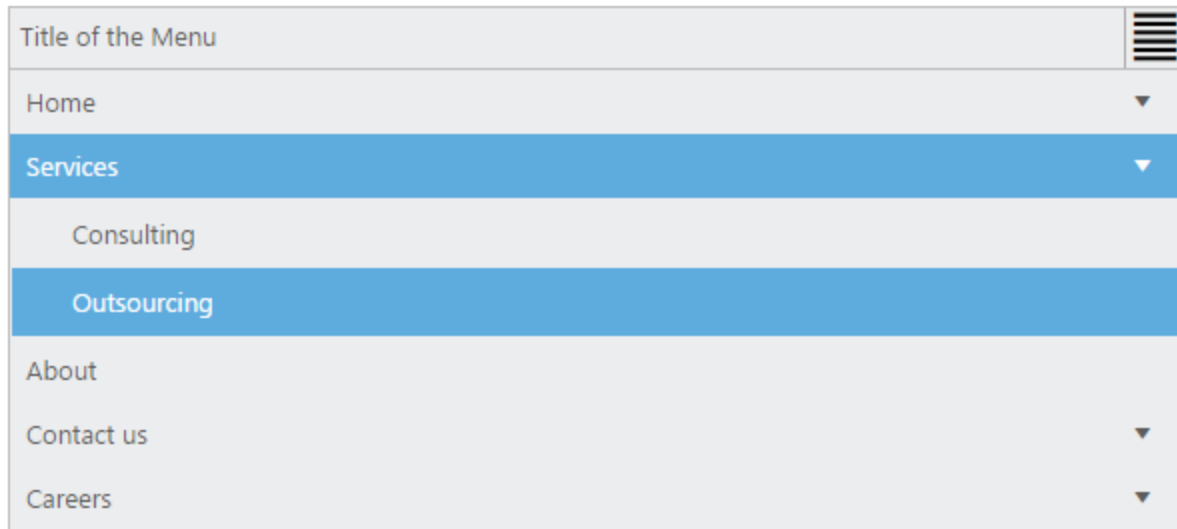
You can specify the title of the **Menu** control in the script as follows.

## JAVASCRIPT

```
$(function () {  
  var sample = new ej.Menu($("#menu"), {  
    titleText: "Title of the Menu"  
  });  
});
```

The following screenshot displays the output of the above code.





#### Show root level arrows

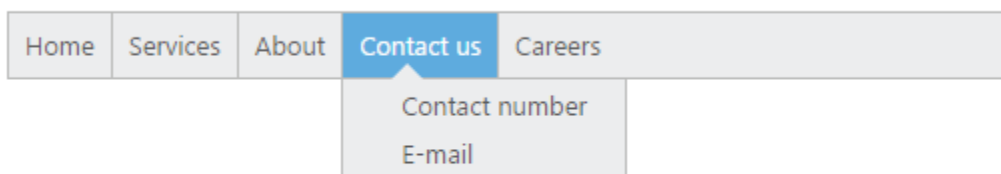
Specifies the main menu item arrows to display only when it contains child menu items. You can use “**showRootLevelArrows**” property to display the arrows of main menu items only when it contains child menu items. This property accepts Boolean value. Its default value is true.

Add the following **<script>** in the above code sample.

#### JAVASCRIPT

```
$(function () {  
    var sample = new ej.Menu($("#menu"), {  
        width: 500,  
        showRootLevelArrows: false  
    });  
});
```

The following screenshot displays the output of the above code.



#### Show sub level arrows

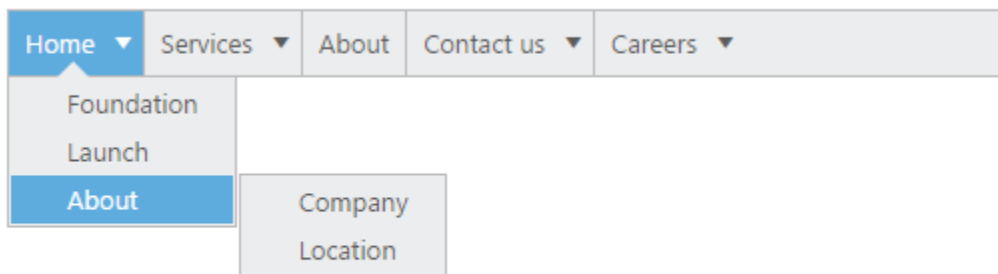
Specifies the sub menu items arrows to display only when it contains child menu items. You can use “**showSubLevelArrows**” property to show the arrows of sub menu items only when it contains child menu items. This property accepts Boolean value. Its default value is true.

Add the following **<script>** in the sample code.

### JAVASCRIPT

```
$(function () {  
  var sample = new ej.Menu($("#menu"), {  
    width: 500,  
    showSubLevelArrows: false  
  });  
});
```

The following screenshot displays the output of the above code.



## Navigation Drawer

### Overview

The Typescript Navigation Drawer control is a sliding panel, which displays the list of view options. By default, the list of view options is not visible, but you can display it onto the left/right side of the window screen by clicking with the desired target icon.

### Key Features

- **Direction:** To change the list view options from the right or left side of the window screen for specifying the direction of the Navigation Drawer.
- **Type:** To change the transition for specifying the type of the Navigation Drawer.
- **Consider SubPage:** Specifies whether the Navigation Drawer to be rendered inside the closest sub page or not.
- **Position:** Specifies the position of the Navigation Drawer whether it's fixed or relative to the page.

### Getting Started

This section helps you to understand the getting started of the Navigation Drawer control with the step-by-step instructions.

#### Create a Navigation Drawer

The following steps guide you to add a Navigation Drawer control.

1) Refer the common Typescript [Getting Started Documentation](#) to create an application and add necessary scripts and styles for rendering the Navigation Drawer control. 2) Create an HTML page and add the scripts and css references in the order mentioned in the following code example.

### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Typescript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.web
.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add Navigation Drawer code here-->
</body>
</html>
```

3) To display the navigation list item by using items property and also you can access the ListView control properties by enabling the enableListView property. Add the following code with in the tag to render the Navigation Drawer with the list view items.

### HTML

```
<div id="navigationPane">
<ul>
<li data-ej-text="Home"></li>
<li data-ej-text="People"></li>
<li data-ej-text="Profile"></li>
</ul>
</div>
```

Create the target element as follows to display the list items by clicking target icon.

### HTML

```
<div id="container">
<div class="e-lv">
<div class="e-header">
<div id="butDrawer"
class="drawerIcon e-icon">
</div>
</div>
</div>
</div>
```

Initialize the Navigation Drawer in ts file by using the ej.NavigationDrawer method.

**TS**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module NavigationDrawerComponent {
$(function () {
var navigationdrawerInstance = new ej.NavigationDrawer($("#navigationPane"),
{
type: "overlay",
direction: "left",
enableListView: true,
listViewSettings: {
width: 300,
selectedIndex: 0
},
position: "normal"
});
});
}
```

To set the target icon image and with the correct position as using the below mentioned styles .

**CSS**

```
<style>
.drawerIcon {
background-position: center center;
background-repeat: no-repeat;
height: 32px;
width: 32px;
background-size: 100% 100%;
padding-right: 10px;
}
.drawerIcon:before {
content: "\e76b";
font-size: 28px;
height: 26px;
}
</style>
```



To open the list items by clicking on target element using the targetId property.

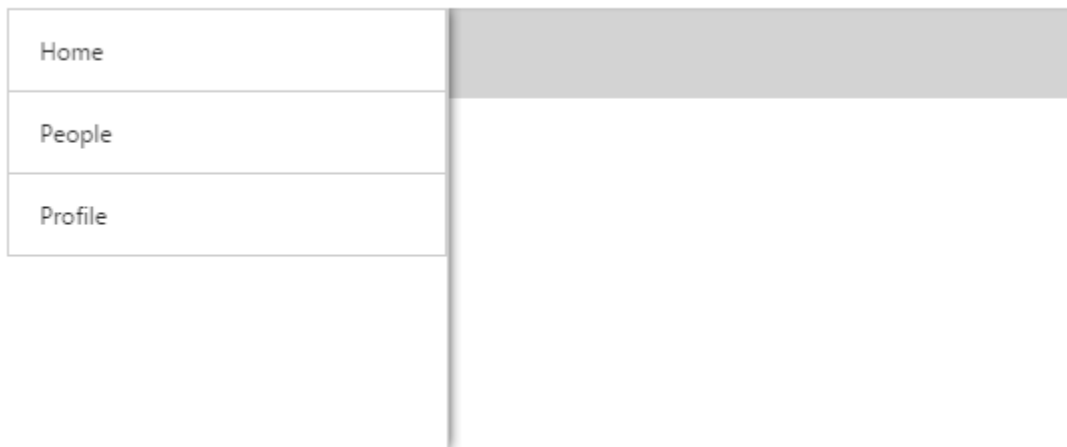
**TS**

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module NavigationDrawerComponent {
$(function () {
```

```

var navigationdrawerInstance = new ej.NavigationDrawer($("#navigationPane"),
{
  targetId: "butDrawer",
  type: "overlay",
  direction: "left",
  enableListView: true,
  listViewSettings: {
    width: 300,
    selectedItemIndex: 0
  },
  position: "normal"
});
});
}

```



To set the images for list items of the Navigation Drawer by using the `[data-ej-imageclass]` property as follows.

### HTML

```

<div id="navigationPane">
  <ul>
    <li data-ej-imageclass="e-icon e-home" data-ej-text="Home"></li>
    <li data-ej-imageclass="e-icon e-photo" data-ej-text="Photos"></li>
    <li data-ej-imageclass="e-icon e-profile" data-ej-text="Profile"></li>
  </ul>
</div>

```

You can set the images with the correct position by using the mentioned styles.

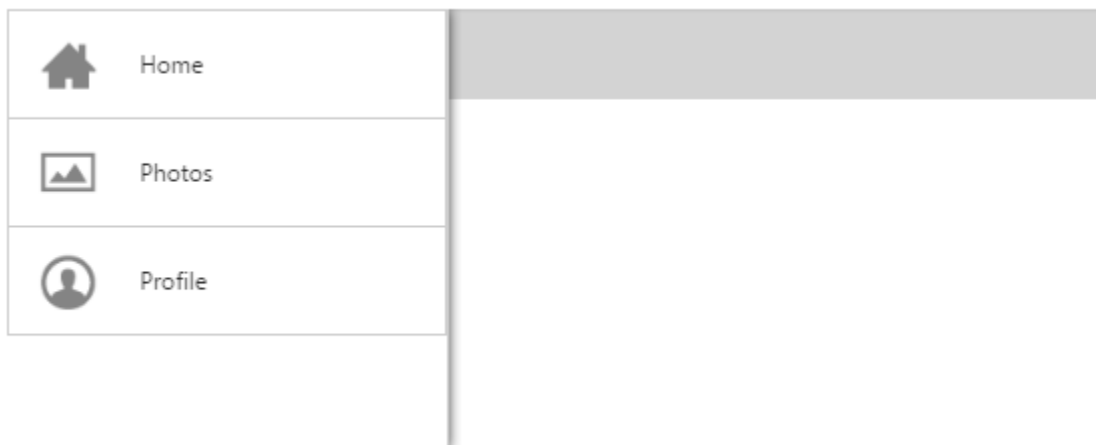
### CSS

```

<style>
@font-face {
  font-family: 'ej-font';
  src: url('../common-images/tools/icons.eot');
  src: url('../common-images/tools/icons.eot') format('embedded-opentype'),
  url('../common-images/tools/icons.woff')

```

```
format('woff'),url('../common-images/tools/icons.woff') format('woff'),
url('../common-images/tools/icons.ttf') format('truetype'),
url('../common-images/tools/icons.svg') format('svg');
font-weight: normal;
font-style: normal;
}
.e-home:before {
font-family: "ej-font";
content: "\e900";
}
.e-profile:before {
font-family: "ej-font";
content: "\e901";
}
.e-photo:before {
font-family: "ej-font";
content: "\e903";
}
.e-location:before {
font-family: "ej-font";
content: "\e905";
}
.e-people:before {
font-family: "ej-font";
content: "\e902";
}
.e-communities:before {
font-family: "ej-font";
content: "\e904";
}
.e-home, .e-profile, .e-people, .e-photo, .e-communities, .e-location {
font-size: 24px;
color: black;
}
</style>
```




**Note:** You can find the Navigation Drawer control properties from the [API reference](#).

## Animations

You can set the transition type of the **Navigation Drawer** by using **type** property. The possible transition types are **slide** and **overlay**.

- **Slide** - both navigation panel and content page slides towards left/right direction to view the navigation panel items.
- **Overlay** - Only the navigation panel slides over the content page to view the navigation panel items. That is, part of the content page is hidden under navigation panel.

---

**Note:**  *Transition slide type works only with fixed position.*

---

The default value is Overlay.

## HTML

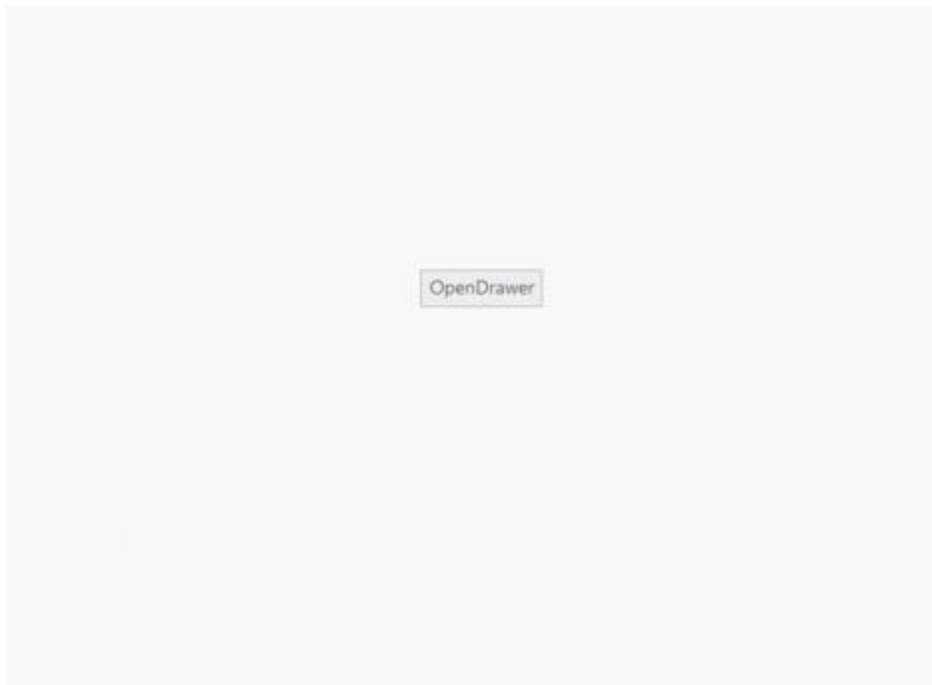
```
<div id="main" style="height:1000px;">
<div id="navpane">
<ul>
<li>Settings</li>
<li>Read</li>
<li>Help</li>
<li>About</li>
</ul>
</div>
<button id="drawerTarget"
style="top:200px;left:600px;position:absolute"></button>
</div>
```

Add the following code in the **script** tag.

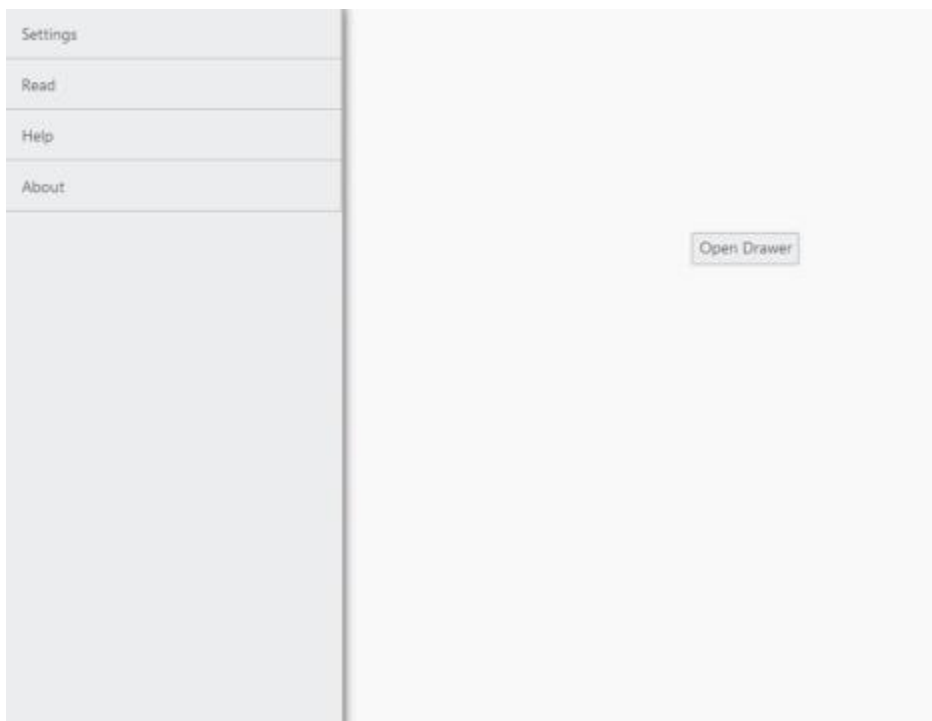
## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module NavigationDrawerComponent {
$(function () {
var navigationdrawerInstance = new ej.NavigationDrawer($("#navpane"), {
type: "slide",
position: "fixed",
targetId: "drawerTarget",
enableListView: true,
listViewSettings: { width: 300 }
});
});
var buttonInstance = new ej.ejButton($("#navpane"), {
text: "OpenDrawer"
});
}
```

The following screenshot illustrates the output.



Before target click



After target click

#### [Customize Direction](#)

By using this property you can set the drawer to be open from right to left direction or left to right direction. The possible direction values are Right, Left and the default value is Left. Refer to the following code example.



## HTML

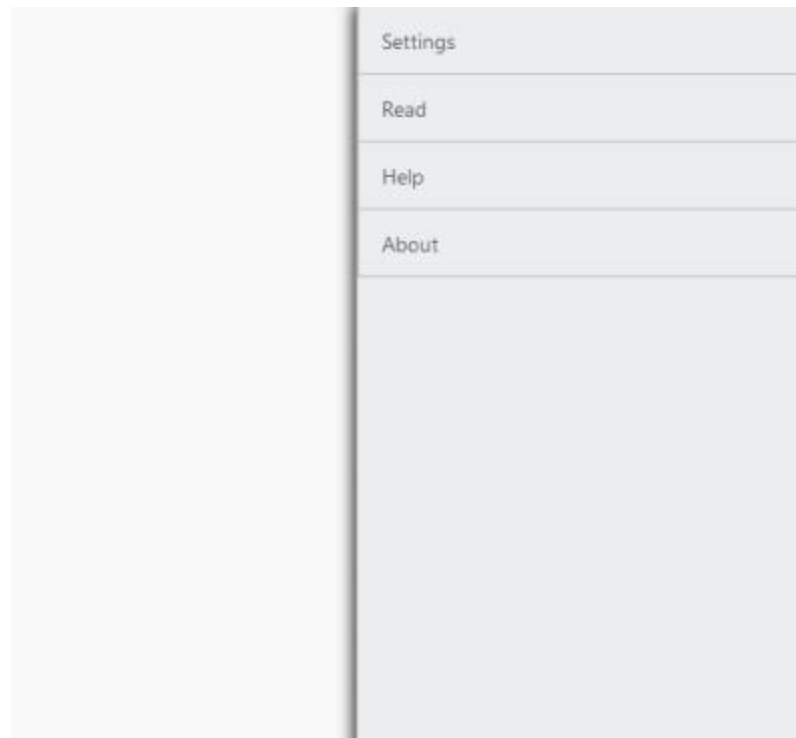
```
<div id="main" style="height:700px;">
<div id="navpane">
<ul>
<li>Settings</li>
<li>Read</li>
<li>Help</li>
<li>About</li>
</ul>
</div>
</div>
```

Add the following code in the **script** tag.

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module NavigationDrawerComponent {
$(function () {
var navigationdrawerInstance = new ej.NavigationDrawer($("#navpane"), {
direction: "right",
position: "fixed",
enableListView: true,
listViewSettings: { width: 300 }
});
});
}
```

The following screenshot displays the output by swiping from right to left at the right side end of the screen.



### Customize Position

**Position** property is used to specify the position whether it is in fixed or relative to the page. When you set a normal value to **position** property, it appears within the container. Otherwise, it appears in the whole body. The possible position values are fixed and normal. The Default value is normal.

### HTML

```
<div id="main" style="height:700px;">
<div id="navpane">
<ul>
<li>Settings</li>
<li>Read</li>
<li>Help</li>
<li>About</li>
</ul>
</div>
</div>
```

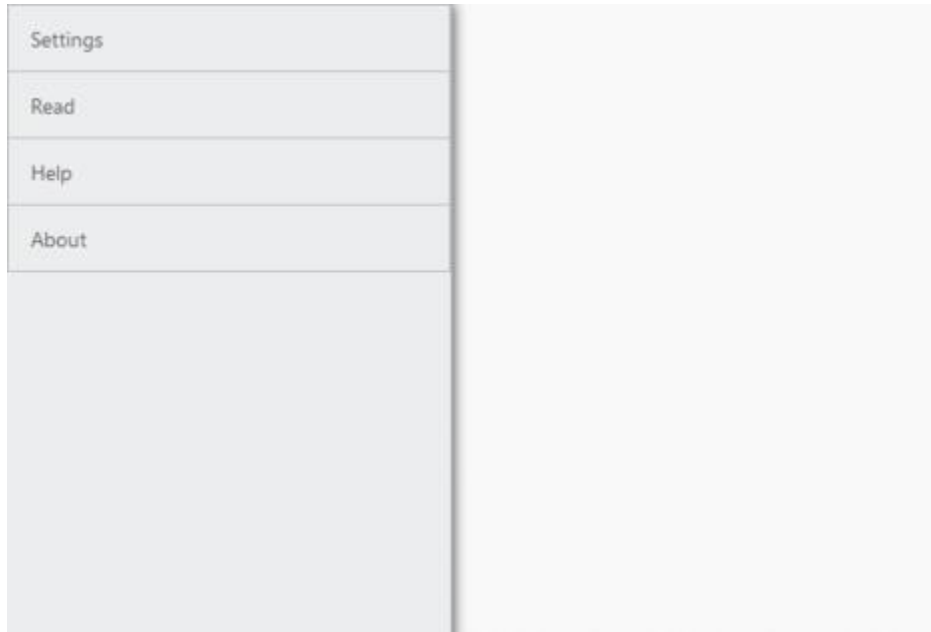
Add the following code in the **script** tag.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module NavigationDrawerComponent {
$(function () {
var navigationdrawerInstance = new ej.NavigationDrawer($("#navpane"), {
position: "fixed",
enableListView: true,
listViewSettings: { width: 300 }
});
});
```

```
});  
}
```

The following screenshot illustrates the output by swiping from left to right at the left end of the screen.



### TargetId

This property is used to define the target Id for **Navigation Drawer**. The drawer opens while you click on the specified target element.

### HTML

```
<button id="drawerTarget"  
style="top:200px;left:600px;position:absolute"></button>  
<div id="navpane">  
  <ul>  
    <li>Settings</li>  
    <li>Read</li>  
    <li>Help</li>  
    <li>About</li>  
  </ul>  
</div>
```

Add the following code in the **script** tag.

### JAVASCRIPT

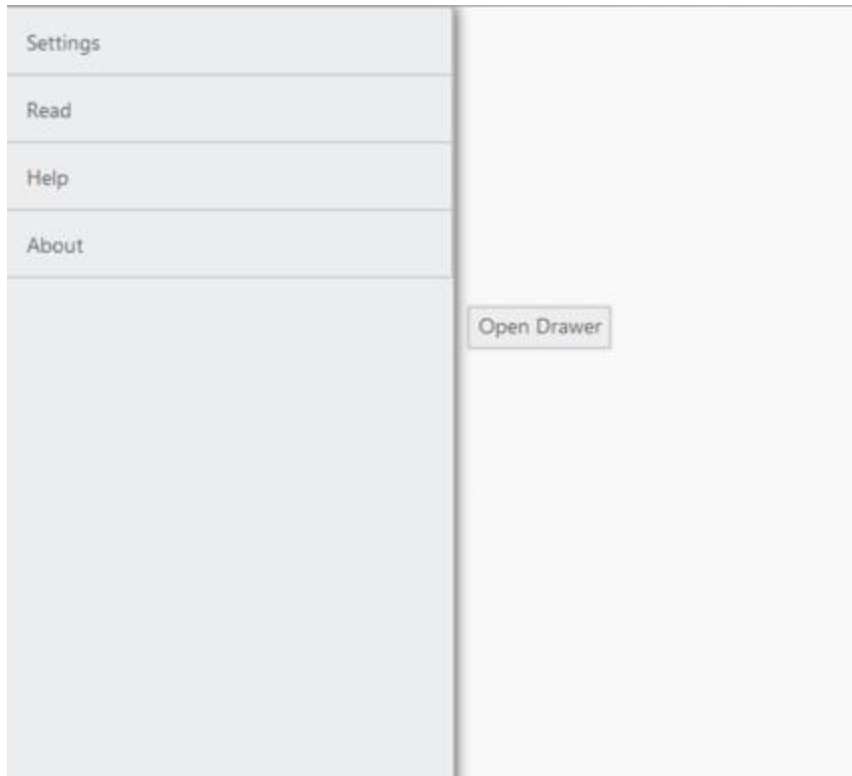
```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module NavigationDrawerComponent {  
  $(function () {  
    var navigationdrawerInstance = new ej.NavigationDrawer($("#navpane"), {  
      position: "fixed",  
      targetId: "drawerTarget",  
      enableListView: true,  
      listViewSettings: { width: 300 }  
    });  
  });  
}
```

```
});  
var buttonInstance = new ej.ejButton($("#drawerTarget"), {  
  text: "Open Drawer"  
});  
});  
}
```

The following screenshots illustrates the output.



Before target click



After target click

## NumericTextbox

### Overview

**Essential JavaScript NumericTextBox** is used to display only numeric values. It has Spin buttons to increase or decrease the values in the Text Box.

### Key Features

- **Min and Max Values** — Specifies value range for the NumericTextBox.
- **Spin Buttons** — Allows to increase or decrease the current value in the NumericTextBox.
- **Step Value** — Allows to increment or decrement the current value by step value.
- **Globalization** — Essential JavaScript NumericTextBox provide **Globalization** support. This control use ej.globalize.js file to globalize the number format, and parse numbers according to the culture.
- **Keyboard Navigation** — Allows to interact with NumericTextBox by using keyboard.
- **RTL Support** — Support for right to left alignment of NumericTextBox input.
- **Decimal Values** — specifies the number of digits to allow after the decimal point in the NumericTextBox.
- **Themes** — Essential JavaScript NumericTextBox consist of 17 built-in themes , and also support custom skins for creating user-defined themes.

### Getting Started

Using the following steps, you can create a **Typescript** NumericTextbox component.

### Creating an NumericTextbox in TypeScript

You can create a **Typescript** application with the help of the given [Typescript Getting Started Documentation](#).

Within an index.html file and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>TypeScript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.web
.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add Textbox sample here-->
</body>
</html>
```

The NumericTextbox can be created from a `input` element with the HTML `id` attribute and pre-defined options set to it.

#### HTML

```
<input id="numeric" type="text" />
<script src="app.js"></script>
```

- Create app.ts file and use the below content

#### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
var number = new ej.NumericTextbox($("#numeric"), {
value: 30,
minValue: 1,
maxValue: 100,
name: "numeric",
width: "100%"
});
}
```

- Now build your application, so that the **app.ts** file will compiled and automatically generated the **app.js** file which is added to your project (User have nothing to do with this file). Now, whatever

code changes that you make in **app.ts** file will be reflected in app.js file by compiling build the application.

Execution of above code will render the following output.

set min and max values

- To set the maximum/ending value of the Textbox, you can use the **maxValue** property. Data type of this property is "number".

**JS**

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var number = new ej.NumericTextbox($("#numeric"), {
    value: 30,
    maxValue: 100,
    name: "numeric",
    width: "100%"
  });
}
```

- To set the minimum/starting value of the Textbox, you can use the **minValue** property. Data type of this property is "number".

**JS**

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module SliderComponent {
  $(function () {
    var number = new ej.NumericTextbox($("#numeric"), {
      value: 30,
      minValue: 1,
      maxValue: 100,
      name: "numeric",
      width: "100%"
    });
  });
}
```

## Behavior Settings

### Decimal Places

The **decimalPlaces** property specifies number of values allowed after the decimal point. The default value of **decimalPlaces** property is 0. i.e., By default you cannot specify decimal value in

NumericTextbox. We need to add this property to allow decimal values. To set the decimalPlaces to “-1”, that allows the decimals without any limit in NumericTextbox control.

#### [Configure Decimal Places](#)

The following steps explain the implementation of **decimalPlaces** in **NumericTextbox**.

In the HTML page set the corresponding <input> elements for rendering NumericTextbox control.

#### **HTML**

```
<input id="numeric" type="text" />
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
    $(function () {
        var number = new ej.NumericTextbox($("#numeric"), {
            value: 333,
            decimalPlaces: 3
        });
    });
}
```

The output for **NumericTextbox** with **decimalPlaces** is as follows.



#### [Persistence Support](#)

The **NumericTextbox** widget provides the state maintenance support. You can maintain the previous changes made in the control after a page refresh.

#### [Configure Persistence Support](#)

The following steps explain the implementation of **enablePersistence** in **NumericTextbox**.

In the HTML page set the corresponding <input> elements for rendering NumericTextbox control.

#### **HTML**

```
<input id="numeric" type="text" />
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
    $(function () {
        var number = new ej.NumericTextbox($("#numeric"), {
            value: 11,
            enablePersistence: true
        });
    });
}
```



```
}

```

The output for **NumericTextbox** with **enablePersistence** is as follows. You can change the value of **NumericTextbox** and reload the web page.

NumericTextbox at initial load

NumericTextbox after changing the value and after page refresh

#### Strict Mode Support

NumericTextbox allows you to use the strict mode option by setting the **enableStrictMode** property. You can set the **minValue** and **maxValue** to the controls to enable strict mode functionality. Default value of this property is **false**. When the textbox value exceeds the **maxValue**, it restricts the exceeded value and returns the **maxValue**. Likewise when the textbox value goes below **minValue**, it restricts the new value and returns the **minValue**. When this property is true, it will not restrict the specified value and an error class will be added to indicate wrong value is provided to the NumericTextbox.

#### Configure Strict Mode Support

The following steps explain the implementation of **enableStrictMode** in **NumericTextbox**.

In the HTML page set the corresponding `<input>` elements for rendering NumericTextbox control.

#### HTML

```
<input id="numeric" type="text" />

```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
    $(function () {
        var number = new ej.NumericTextbox($("#numeric"), {
            value: 10, //value(10) exceeds maxValue(5), so it will add the error class.
            minValue: -3,
            maxValue: 5,
            enableStrictMode: true
        });
    });
}
```

The output for NumericTextbox when **enableStrictMode** is "true" is as follows.



### Enabled or Disabled

The **NumericTextBox** control has an option to enable or disable its element. You can set the **enabled** property as “**true**” to enable the NumericTextBox control.

Also you can enable/disable the **NumericTextBox** by using [enable](#) and [disable](#) methods.

### Configure Enabled or Disabled

The following steps explain the implementation of **enabled** in **NumericTextBox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **NumericTextBox** control.

### HTML

```
<input id="numeric" type="text" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
    $(function () {
        var number = new ej.NumericTextbox($("#numeric"), {
            value: 1,
            enabled: false
        });
    });
}
```

The output for NumericTextBox when enabled is “false” and when enabled is “true”.



NumericTextBox with enabled as false



NumericTextBox with enabled as true

### Adjusting Textbox Size

The **NumericTextBox** size can be modified by using the **height** and **width** properties. You can customize the size of NumericTextBox by using these properties.

### Configure Height and Width

The following steps explain the implementation of **height** and **width** in NumericTextBox .

In the HTML page set the corresponding <input> elements for rendering NumericTextBox control.

### HTML

```
<input id="numeric" type="text" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
    $(function () {
        var number = new ej.NumericTextbox($("#numeric"), {
            width: 100, height: 50, value: 1
        });
    });
}
```

The output for NumericTextBox after setting “height” and “width” is as follows.



### Increment Step

The **incrementStep** property is used to increase or decrease the amount of value in the NumericTextBox control.

### Configure Increment Step

The following steps explain the implementation of **incrementStep** in NumericTextBox.

In the HTML page set the corresponding <input> elements for rendering NumericTextBox control.

### HTML

```
<input id="numeric" type="text" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
    $(function () {
        var number = new ej.NumericTextbox($("#numeric"), {
            value: 1,
            incrementStep: 2
        });
    });
}
```

Output of Numeric textbox with **incrementStep** is as follows.



NumericTextBox at initial load



NumericTextBox after increasing two step

#### Define Name

When you have placed the **NumericTextBox** in a form, the **name** property is used to send the field value at form submission. The default value of the **name** property is null.

#### Configure Name

The following steps explain the implementation of **name** in **NumericTextBox**.

In the HTML page set the corresponding `<input>` elements for rendering **NumericTextBox** control.

#### HTML

```
<input id="numeric" type="text" />
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
  $(function () {
    var number = new ej.NumericTextbox($("#numeric"), {
      value: 12,
      name: "numeric"
    });
  });
}
```

#### Define Value

The value of **NumericTextBox** can be assigned by using the **value** property. The default value for **value** property is null.

You can get the value of **NumericTextBox** by using [getValue](#) method.

#### Configure Value

The following steps explain the implementation of **value** in **NumericTextBox**.

In the **HTML** page set the corresponding `<input>` elements for rendering **NumericTextBox** control.

#### HTML

```
<input id="numeric" type="text" />
```

**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.NumericTextbox($("#numeric"), {
value: 12
});
});
}

```

The output for **NumericTextBox** with the **value** property is as follows.



Define **maxValue** and **minValue**

*maxValue*

The maximum limit value can be assigned to the **NumericTextBox** by using the **maxValue** property. The default value of **maxValue** property is 1.7976931348623157e+308.

*minValue*

The minimum limit value can be assigned to the **NumericTextBox** by using the **minValue** property. The default value of **minValue** property is -1.7976931348623157e+308.

*Configure maxValue and minValue*

The following steps explain the implementation of **maxValue** and **minValue** in **NumericTextBox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **NumericTextBox** control.

**HTML**

```

<input id="numeric" type="text" />

```

**JAVASCRIPT**

```

//NumericTextBox with maxValue
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.NumericTextbox($("#numeric"), {
maxValue: 2,
value:3
});
});
}

//NumericTextBox with minValue
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.NumericTextbox($("#numeric"), {

```

```
minValue: -1,  
value:-2  
});  
});  
}
```

The output for **NumericTextbox** with basic properties is as follows.



NumericTextbox with **maxValue**



NumericTextbox with **minValue**

#### Read Only Support

The **NumericTextbox** supports read only option. When you enable the **readOnly** property to the control, the value cannot be changed in the NumericTextbox . You can set the **readOnly** property as “true” to enable this option.

#### Configure Read Only

The following steps explain the implementation of **readOnly** in **NumericTextbox** .

In the **HTML** page set the corresponding **<input>** elements for rendering **NumericTextbox** control.

#### HTML

```
<input id="numeric" type="text" />
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module EditorComponent {  
    $(function () {  
        var number = new ej.NumericTextbox($("#numeric"), {  
            value: 1,  
            readOnly: true  
        });  
    });  
}
```

The output for NumericTextbox when **readOnly** is “true” is as follows. The NumericTextbox value cannot be edited or changed.



## Appearance

### Theme

The NumericTextBox control style and appearance can be controlled based on CSS classes. In order to apply styles to the NumericTextBox control, you need to refer 2 files namely, **ej.widgets.core.min.css** and **ej.theme.min.css**. If the file **ej.web.all.min.css** is referred, then it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.web.all.min.css** is the combination of these two.

By default, there are 17 themes support available for **Textbox** control namely:

- bootstrap
- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- high-contrast-01
- high-contrast-02
- material
- office-365

### CSS Class

The **CSS** can be customized by using the **cssClass** in the NumericTextBox . You can customize the NumericTextBox with various **cssClass** properties to appear like your desired control.

### Configure CSS Class

The following steps explain the implementation of **cssClass** in NumericTextBox .

In the HTML page set the corresponding `<input>` elements for rendering NumericTextBox controls.

### HTML

```
<input id="numeric" type="text" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
  $(function () {
```

```
var number = new ej.NumericTextbox($("#numeric"), {
  value: 1,
  cssClass: "customCss"
});
```

Customize the CSS properties in custom CSS class.

### CSS

```
<style>
.customCss .e-box {
border-color: #9d241b;
}
.customCss .e-input {
background-color: #f6db8d;
}
.customCss .e-select {
background-color: #ecf6ac;
border-color: #3c36e7;
}
</style>
```

The output for NumericTextBox after applying **cssClass** is as follows.



### Rounded Corner Support

The NumericTextBox provides you with rounded corner support whose appearance is different from normal textbox controls.

#### Configure Rounded Corner Support

The following steps explain the implementation of **showRoundedCorner** in **NumericTextBox**.

In the HTML page set the corresponding **<input>** elements for rendering NumericTextBox control.

### HTML

```
<input id="numeric" type="text" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
  $(function () {
    var number = new ej.NumericTextbox($("#numeric"), {
      value: 1,
      showRoundedCorner: true
    });
  });
}
```



```
}
```

The output for NumericTextBox when **showRoundedCorner** is “true”.



### Spin Button Support

The NumericTextBox provides you the option as to whether to display the spin button in the widget or remove it from the control by using **showSpinButton** property.

#### Configure Spin Button

The following steps explain the implementation of **showSpinButton** in **NumericTextBox**.

In the HTML page set the corresponding <input> elements for rendering NumericTextBox controls.

#### HTML

```
<input id="numeric" type="text" />
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
  $(function () {
    var number = new ej.NumericTextbox($("#numeric"), {
      value: 1,
      showSpinButton: false
    });
  });
}
```

The output for NumericTextBox when showSpinButton is “false”.



### Water Mark Text Support

The NumericTextBox provide water mark text support. You can display the initial value in the control by water mark.

#### Configure Water Mark Text

The following steps explain the implementation of **watermarkText** in **NumericTextBox**.

In the HTML page set the corresponding <input> elements for rendering NumericTextBox control.

#### HTML

```
<input id="numeric" type="text" />
```

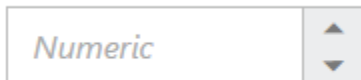
**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.NumericTextbox($("#numeric"), {
watermarkText: "Numeric"
});
});
}

```

The output for NumericTextBox after applying **watermarkText** is as follows.

**Globalization Support**

**Globalization** is language support based on the culture in **NumericTextBox**. You can achieve the **Globalization** using "**locale**" property in **NumericTextBox**.

The widget provides multi-language support using globalization. You can customize the NumericTextBox with your own language style by using this feature. You can change the globalization by using the **locale** property. The default value for **locale** property is **en-US** in NumericTextBox control.

More than 350 culture specific files are available to localize the value. To know more about EJ globalize support, please refer the below link

<https://help.syncfusion.com/js/localization>

**Note:** All the culture-specific script files are available within the below specified location, once you have installed Essential Studio in your machine, therefore it is not necessary to download these files explicitly.

```
'(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n'
```

For example, If you have installed the Essential Studio package within C:\Program Files (x86), then navigate to the below location, C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

Refer the below German culture file in head section of HTML page after the reference of **ej.web.all.min.js** file.

**JAVASCRIPT**

```

<script src="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/i18n/ej.culture.de-DE.min.js"></script>

```

You can dynamically change the language based on their culture.

**Configure Globalization**

The following example describes the way to use Globalization for NumericTextBox widget.

**HTML**

```
<input id="numeric" type="text" />
```

**JAVASCRIPT**

```
/* Numeric Textbox */
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.NumericTextbox($("#numeric"), {
value: 12345,
decimalPlaces: 2,
locale: "de-DE"
});
});
}
```

The output for **NumericTextBox** with Globalization.



NumericTextBox with de-DE locale



NumericTextBox with en-US locale

**RTL Support**

The **NumericTextBox** provides **RTL (Right-To-Left)** support. The alignment of NumericTextBox can be changed from **Left-To-Right** into **Right-To-Left**.

**Enable RTL**

The following steps explain the implementation of **enableRTL** in **NumericTextBox**.

In the HTML page set the corresponding <input> elements for rendering NumericTextBox control.

**HTML**

```
<input id="numeric" type="text" />
```

**JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
```

```
var number = new ej.NumericTextbox($("#numeric"), {
  value: 11,
  enableRTL: true
});
});
}
```

The output for **NumericTextBox** when **enableRTL** is “true” is as follows.



### Keyboard Interaction

With the keyboard navigation enabled in the **NumericTextBox** control, it is possible to control the actions of the **NumericTextBox** with the provided shortcut keys. Almost all the **NumericTextBox** actions that are done through mouse can be controlled with shortcut keys.

The various keyboard shortcuts available within the **NumericTextBox** control are discussed in the following table.

#### Keyboard Shortcuts

| Shortcut Key          | Description            |
|-----------------------|------------------------|
| <u>Access key</u> + j | Focuses the control    |
| Up                    | Increments the value   |
| Down                  | Decrements the value   |
| Tab                   | Focus the next element |

### Configuring Keyboard Navigation

The following steps explain the implementation of keyboard interaction in **NumericTextBox**.

In the HTML page set the corresponding <input> elements for rendering NumericTextBox controls. Set the access key property to the NumericTextBox control for focusing the control while key is pressed.

#### HTML

```
<input id="numeric" type="text" accesskey="j"/>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
  $(function () {
    var number = new ej.NumericTextbox($("#numeric"), {
      value: 11
    });
  });
}
```

```
}};  
}
```

Run the above example and press [Access key](#) + **j** key to focus the **NumericTextBox** widget. Perform provided functionality by using keyboard shortcuts.



## How To

### Use Normal Textboxes as Syncfusion textboxes

In an application or a web page, you may use normal textboxes along with other Syncfusion components, since there is no separate Essential JavaScript plugin for textboxes.

So, you may want to make a normal textbox to look like Syncfusion textbox in order to achieve uniform look and appearance in your web page.

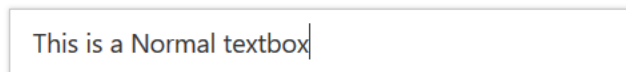
This can be achieved by adding “**e-textbox**” class to the HTML element.

By adding this class, the textbox will have standard look and appearance as other components for all the themes supported by Syncfusion.

### HTML

```
<input type="text" class="e-textbox" value="This is a normal Textbox"/>
```

Textbox will be rendered as shown below.



Normal textbox as Syncfusion textbox

## PDF viewer

### Overview

PDF viewer for JavaScript is a visualization component to view PDF documents in web pages. It is powered by HTML5/JavaScript and provides various customization.

### Key Features

The key features of the **PDF viewer** control are listed as follows:

- Supports uploading PDF document into the server as byte array or as stream and supports displaying the same in the client.
- Supports for viewing password protected/encrypted PDF documents.
- Supports customizing the default toolbar or creation of a toolbar as per the application requirements.
- Supports zooming tools and viewing modes for better viewing experience.

- Supports built-in themes for enhancing appearance.
- Supports printing the PDF document.
- Supports responsive rendering while resizing the control/window.
- Supports downloading the PDF document being displayed.
- Compatible with all the modern browsers that provides support for HTML5/CSS3/JavaScript.

## Getting Started

This section explains briefly about how to integrate a PDF viewer control in your TypeScript web application.

For common steps to create a project in typescript, you can refer [here](#).

The default type definition file ej.web.all.d.ts needs to be included the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the jquery.d.ts file in your TypeScript application, which can be downloaded from [here](#).

## Script and CSS Reference

Add the scripts and CSS references to the “index.html” page in the order mentioned in the following code example.

### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
<script
src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.validate.
min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

**Note:** 1. In production, we highly recommend you to use our [custom script generator](#) to create custom script file with required controls and its dependencies only. Also to reduce the file size further please use [GZip compression](#) in your server.

**Note:** 2. For themes, you can use the `ej.web.all.min.css` CDN link from the code snippet given. To add the themes in your application, please refer to [this link](#).

### Control Initialization

Add a `div` container to render the PDF viewer control.

### HTML

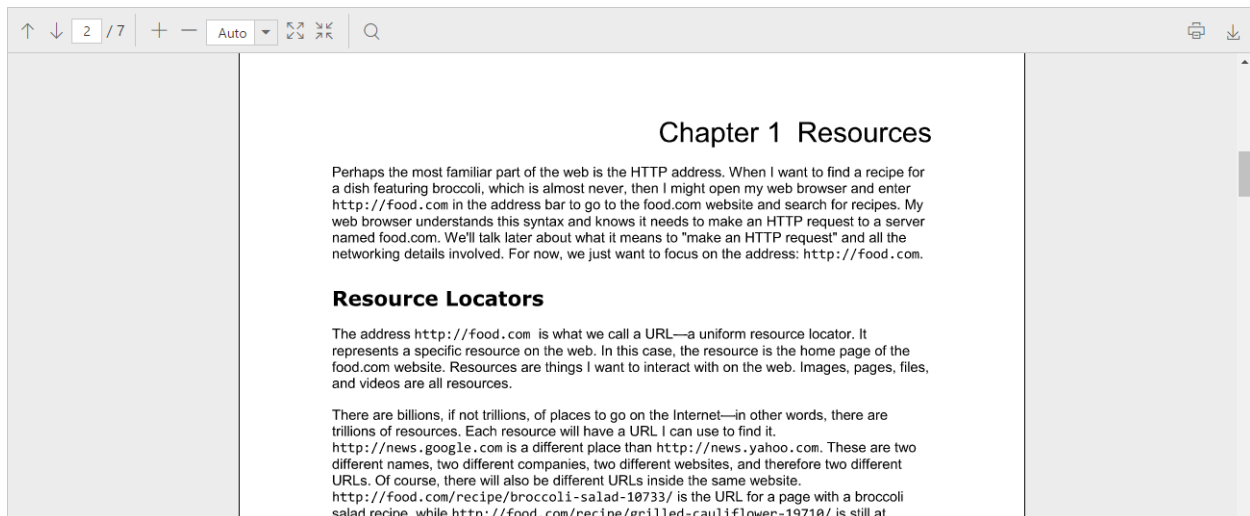
```
<!DOCTYPE html>
<html>
<body>
<div id="pdfviewer1"></div>
</body>
</html>
```

Initialize the PDF viewer in `app.ts` file by using the `ej.PdfViewer` method.

### HTML

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module PdfViewerComponent {
$(function () {
var viewer = new ej.PdfViewer($("#pdfviewer1"), {
serviceUrl: "http://js.syncfusion.com/ejservices/api/PdfViewer",
isResponsive: true
});
});
}
```

**Note:** Default PDF document will be rendered in the PDF viewer control, which is used in the online service.



## PercentageTextbox

### Overview

**Essential JavaScript PercentageTextBox** control is used to display only percentage values. It has Spin buttons to increase or decrease the values in the Text Box.

### Key Features

- **Min and Max Values** — Specifies value range for the Text Box.
- **Spin Buttons** — Allows you to increase or decrease the current value in the Text Box.
- **Step Value** — Allows you to increment or decrement the current value by step value.
- **Globalization** — Essential JavaScript Text Boxes provide globalization support. This control use `ej.globalize.js` file to globalize the number format, and parse numbers according to the culture.
- **Keyboard Navigation** — You can interact with Text Boxes by using keyboard.
- **RTL Support** — Support for right to left alignment of Text Box input.
- **Decimal Values** — You can configure Text Boxes to accept decimal values.
- **Themes** — **Essential JavaScript PercentageTextBox** consist of 17 built-in themes , and also support custom skins for creating user-defined themes.

### Getting Started

Using the following steps, you can create a **TypeScript** PercentageTextbox component.

#### Creating an PercentageTextbox in TypeScript

You can create a **TypeScript** application with the help of the given [TypeScript Getting Started Documentation](#).

Within an index.html file and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>TypeScript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.web
.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add Textbox sample here-->
</body>
</html>
```

The PercentageTextbox can be created from a `input` element with the HTML `id` attribute and pre-defined options set to it.

#### HTML



```
<input id="percent" type="text" />
<script src="app.js"></script>
```

- Create app.ts file and use the below content

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var per = new ej.PercentageTextbox($("#percent"), {
    value: 60,
    name: "percent",
    width: "100%"
  });
}
```

- Now build your application, so that the **app.ts** file will be compiled and automatically generated the **app.js** file which is added to your project (User have nothing to do with this file). Now, whatever code changes that you make in **app.ts** file will be reflected in app.js file by compiling build the application.

Execution of above code will render the following output.



### set min and max values

- To set the maximum/ending value of the Textbox, you can use the **maxValue** property. Data type of this property is "number".

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module EditorComponent {
  var per = new ej.PercentageTextbox($("#percent"), {
    value: 60,
    maxValue: 1000,
    name: "percent",
    width: "100%"
  });
}
```

- To set the minimum/starting value of the Textbox, you can use the **minValue** property. Data type of this property is "number".

### JS

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var per = new ej.PercentageTextbox($("#percent"), {
value: 60,
minValue: 10,
maxValue: 1000,
name: "percent",
width: "100%"
});
});
}

```

## Behavior Settings

### Decimal Places

The property **decimalPlaces** declares the decimal point to the value of **PercentageTextBox** control. The default value of **decimalPlaces** is 0 in **PercentageTextBox** control. To set the decimalPlaces to "-1", that allows the decimals without any limit in PercentageTextBox control.

#### Configure Decimal Places

The following steps explain the implementation of **decimalPlaces** in **PercentageTextBox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextBox** control.

### HTML

```

<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>

```

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value: 22,
decimalPlaces: 2
});
});
}

```

The output for **PercentageTextBox** with **decimalPlaces** is as follows.

Percent

### Persistence Support

The **PercentageTextbox** widget provides the state maintenance support. You can maintain the previous changes made in the control after a page loads.

### Configure Persistence Support

The following steps explain the implementation of **enablePersistence** in **PercentageTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextbox** control.

### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>
```

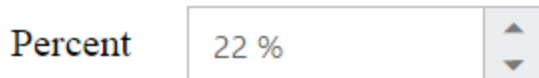
### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value: 22,
enablePersistence: true
});
});
}
```

The output for **PercentageTextbox** with **enablePersistence** is as follows. You can change the value of **PercentageTextbox** and reload the web page.

Percent

PercentageTextbox at initial load



PercentageTextbox after changing the value and a page load

#### Strict Mode Support

The **PercentageTextbox** widget allows you to use the strict mode option by setting the **enableStrictMode** property. You can set the **minValue** and **maxValue** to the controls to enable strict mode functionality. When the Textbox value exceeds the **maxValue**, it restricts the exceeded value and returns the **maxValue**. Likewise when the Textbox value goes below **minValue**, it restricts the new value and returns the **minValue**. When this property is true, it will not restrict the specified value and an error class will be added to indicate wrong value is provided to the PercentageTextbox.

#### Configure Strict Mode Support

The following steps explain the implementation of **enableStrictMode** in **PercentageTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextbox** control.

#### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value: -10, //value(-10) is under minValue(-5), so Error class will be added.
minValue: -5,
maxValue: 3,
enableStrictMode: true
});
});
}
```

The output for **PercentageTextbox** when **enableStrictMode** is **"true"** is as follows.

Percent

### Enabled or Disabled

The **PercentageTextbox** control has an option to enable or disable its element. You can set the **enabled** property as "true" to enable the **PercentageTextbox** control.

Also you can enable/disable the **PercentageTextbox** by using [enable](#) and [disable](#) methods.

### Configure Enabled or Disabled

The following steps explain the implementation of **enabled** in **PercentageTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextbox** control.

### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value: 2,
enabled: false
});
});
}
```

The output for **PercentageTextbox** when **enabled** is "true" and when **enabled** is "false".

Percent

PercentageTextbox with enabled as true

Percent

PercentageTextbox with enabled as false

#### Adjusting PercentageTextbox Size

The **PercentageTextbox** size can be modified by using the **height** and **width** properties. You can customize the size of **PercentageTextbox** by using these properties.

#### Configure Height and Width

The following steps explain the implementation of **height** and **width** in **PercentageTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextbox** control.

#### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value: 2,
width: 100,
height: 40
});
});
}
```

The output for **PercentageTextbox** after setting “**height**” and “**width**” is as follows.

Percent

### Increment Step

The **incrementStep** property is used to increase or decrease the amount of value in the **PercentageTextBox** control.

#### Configure Increment Step

The following steps explain the implementation of **incrementStep** in **PercentageTextBox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextBox** control.

#### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>
```

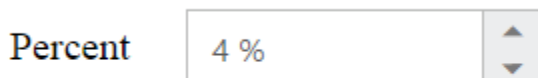
#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value:1,
incrementStep: 3
});
});
}
```

The output for **PercentageTextBox** with **incrementStep** is as follows.



PercentageTextBox at initial load



PercentageTextBox after increasing one step

### Define Name

When you have placed the **PercentageTextbox** in a form, the **name** property is used to send the field value at form submission. The default value of the **name** property is null.

#### Configure Name

The following steps explain the implementation of **name** in **PercentageTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextbox** control.

#### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
name: "percent"
});
});
}
```

### Define Value

The value of **PercentageTextbox** can be assigned by using the **value** property. The default value for **value** property is null.

You can get the value of **PercentageTextbox** by using [getValue](#) method.

#### Configure Value

The following steps explain the implementation of **value** in **PercentageTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextbox** control.

#### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>
```



```

</td>
</tr>
</tbody>
</table>

```

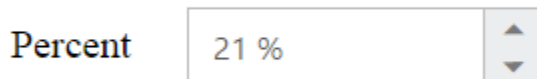
### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value: 21
});
});
}

```

The output for **PercentageTextbox** with the **value** property is as follows.



Define **maxValue** and **minValue**

#### *maxValue*

The maximum limit value can be assigned to the **PercentageTextbox** by using the **maxValue** property. The default value of **maxValue** property is 1.7976931348623157e+308.

#### *minValue*

The minimum limit value can be assigned to the **PercentageTextbox** by using the **minValue** property. The default value of **minValue** property is -1.7976931348623157e+308.

#### *Configure maxValue and minValue*

The following steps explain the implementation of **maxValue** and **minValue** in **PercentageTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextbox** control.

### HTML

```

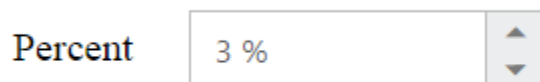
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>

```

**JAVASCRIPT**

```
//PercentageTextbox with maxValue
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
maxValue: 3,
value:5
});
});
}
//PercentageTextbox with minValue
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
minValue: -2,
value:-5
});
});
}
```

The output for **PercentageTextbox** with basic properties is as follows.



PercentageTextBox with maxValue



PercentageTextBox with minValue

**Read Only Support**

The **PercentageTextbox** supports read only option. When you enable the **readOnly** property to the control, the value cannot be changed in the **PercentageTextbox**. You can set the **readOnly** property as **"true"** to enable this option.

**Configure Read Only**

The following steps explain the implementation of **readOnly** in **PercentageTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextbox** control.

**HTML**

```
<table cellpadding="10">
<tbody>
```

```

<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value: 3,
readOnly: true
});
});
}

```

The output for **PercentageTextbox** when **readOnly** is **"true"** is as follows. The **PercentageTextbox** values cannot be edited or changed.



## Appearance

### Theme

The **PercentageTextbox** control's style and appearance can be controlled based on CSS classes. In order to apply styles to the **Textbox** control, you need to refer 2 files namely, **ej.widgets.core.min.css** and **ej.theme.min.css**. If the file **ej.web.all.min.css** is referred, then it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.web.all.min.css** is the combination of these two.

By default, there are 17 themes support available for **PercentageTextbox** control namely:

- bootstrap
- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

- high-contrast-01
- high-contrast-02
- material
- office-365

### CSS Class

The **CSS** can be customized by using the **cssClass** in the **PercentageTextbox**. You can customize the **PercentageTextbox** with various **cssClass** properties to appear like your desired control.

### Configure CSS Class

The following steps explain the implementation of **cssClass** in **PercentageTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextbox** control.

### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value: 2,
cssClass: "customCss"
});
});
}
```

Customize the CSS properties in custom CSS class.

### CSS

```
<style>
.customCss .e-box {
border-color: #9d241b;
}
.customCss .e-input {
background-color: #f6db8d;
}
.customCss .e-select {
background-color: #ecf6ac;
}
```

```
border-color: #3c36e7;
}
</style>
```

The output for **PercentageTextbox** after applying **cssClass** is as follows.

Percent

### Rounded Corner Support

The **PercentageTextbox** provides you with rounded corner support whose appearance is different from normal textbox controls.

### Configure Rounded Corner Support

The following steps explain the implementation of **showRoundedCorner** in **PercentageTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextbox** control.

### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value: 2,
showRoundedCorner: true
});
});
}
```

The output for **PercentageTextbox** when **showRoundedCorner** is "true".

Percent

### Spin Button Support

The **PercentageTextbox** provides you the option as to whether to display the spin button in the widget or remove it from the control by using **showSpinButton** property.

#### Configure Spin Button

The following steps explain the implementation of **showSpinButton** in **PercentageTextbox** .

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextbox** control.

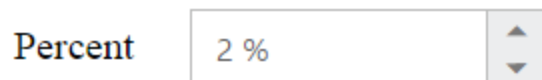
#### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>
```

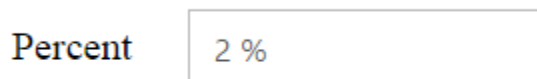
#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value: 2,
showSpinButton: true
});
});
}
```

The output for **PercentageTextbox** when **showSpinButton** is “true”.



PercentageTextbox with showSpinButton is true



PercentageTextbox with showSpinButton is false

### Water Mark Text Support

The **PercentageTextbox** provide water mark text support. You can display the initial value in the control by water mark.

#### Configure Water Mark Text

The following steps explain the implementation of **watermarkText** in **PercentageTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextbox** controls.

#### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
watermarkText: "Percentage"
});
});
}
```

The output for **PercentageTextbox** after applying **watermarkText** is as follows.



### Globalization Support

**Globalization** is language support based on the culture in **Textboxes**. You can achieve the **Globalization** using **"locale"** property in **PercentageTextbox**.

The **PercentageTextbox** widget provides multi-language support using globalization. You can customize the **PercentageTextbox** with your own language style by using this feature. You can change the globalization by using the **locale** property. The default value for **locale** property is **en-US** in **PercentageTextbox** control.

More than 350 culture specific files are available to localize the value. To know more about EJ globalize support, please refer the below link

<https://help.syncfusion.com/js/localization>

**Note:** All the culture-specific script files are available within the below specified location, once you have installed Essential Studio in your machine, therefore it is not necessary to download these files explicitly.

```
'(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n'
```

For example, If you have installed the Essential Studio package within C:\Program Files (x86), then navigate to the below location, C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

Refer the below German culture file in head section of HTML page after the reference of **ej.web.all.min.js** file.

### JAVASCRIPT

```
<script src="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/i18n/ej.culture.de-DE.min.js"></script>
```

You can dynamically change the language based on their culture.

### Configure Globalization

The following example describes the way to use Globalization for **PercentageTextbox** widget.

### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value: 21234,
decimalPlaces: 3,
locale: "de-DE"
});
});
}
```

The output for **PercentageTextbox** with Globalization.



Percent

PercentageTextBox with de-DE locale

Percent

PercentageTextBox with en-US locale

### RTL Support

The **PercentageTextBox** provides **RTL (Right-To-Left)** support. The alignment of **PercentageTextBox** can be changed from **Left-To-Right** into **Right-To-Left**.

### Enable RTL

The following steps explain the implementation of **enableRTL** in **PercentageTextBox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextBox** control.

### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" />
</td>
</tr>
</tbody>
</table>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value: 22,
enableRTL: true
});
});
}
```

The output for **PercentageTextBox** when **enableRTL** is **"true"** is as follows.

Percent

## Keyboard Interaction

With the keyboard navigation enabled in the **PercentageTextbox** control, it is possible to control the actions of the **PercentageTextbox** with the provided shortcut keys. Almost all the **PercentageTextbox** actions that are done through mouse can be controlled with shortcut keys.

The various keyboard shortcuts available within the **PercentageTextbox** control are discussed in the following table.

### Keyboard Shortcuts

| Shortcut Key          | Description                           |
|-----------------------|---------------------------------------|
| <u>Access key</u> + j | Focuses the PercentageTextbox control |
| Up                    | Increments the value                  |
| Down                  | Decrements the value                  |

## Configuring Keyboard Navigation

The following steps explain the implementation of keyboard interaction in **PercentageTextbox**.

In the **HTML** page set the corresponding **<input>** elements for rendering **PercentageTextbox** control. Set the **access key** property to the **PercentageTextbox** control for focusing the control while key is pressed.

### HTML

```
<table cellpadding="10">
<tbody>
<tr>
<td>
<label for="percent">Percent</label>
</td>
<td>
<input id="percent" type="text" accesskey="j" />
</td>
</tr>
</tbody>
</table>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module EditorComponent {
$(function () {
var number = new ej.PercentageTextbox($("#percent"), {
value: 22
});
});
}
```

Run the above example and press [Access key](#) + j key to focus the **PercentageTextbox** widget. Perform provided functionality by using keyboard shortcuts.



## PivotChart

### Overview

The **PivotChart** control is a lightweight control that reads both **OLAP** and **Relational** datasource and visualizes it in graphical format with the ability to drill up and down.

### Key Features

The key features of the **PivotChart** control is listed as follows:

- **Data source** - Supports OLAP data binding with XML/A data sources and Relational data sources.
- **Series** - A collection of series items that contain the actual data points is displayed on the chart.
- **Legend** - A color code that helps to differentiate between chart items. A legend also has labels beside each color to indicate that it applies to information from Series 1, Series 2, and so on.
- **Drill support** - Enables you to navigate to the inner levels in the row axis.
- **Tooltip** - Displays data point values of respective chart series on mouse hover.
- **Zooming and scrolling** – Enables you to zoom into an area of the chart so the data can be viewed with more clarity.

### Getting Started

This section explains you the steps required to populate the PivotChart with data source. This section covers only the minimal features that you need to know to get started with the PivotChart.

For common steps of typescript , you can refer [here](#).

The default type definition file ej.web.all.d.ts needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the jquery.d.ts file in your TypeScript application, which can be downloaded from [here](#).

### Script and CSS Reference

Add the scripts and CSS references to the “index.html” page as the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
```

```
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
<script
src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.valida
te.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

In the above code, `ej.web.all.min.js` script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

### Relational

This section covers the information that you need to know to populate a simple PivotChart with Relational data source.

#### Control Initialization

Add a `div` container to render the PivotChart.

### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="PivotChart1" style="min-height: 275px; min-width: 525px; height:
460px; width: 100%"></div>
</body>
</html>
```

Initialize the PivotChart in `app.ts` file by using the `ej.PivotChart` method.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), { });
});
```

#### Populate PivotChart with Data

Let us now see how to populate the PivotChart control using a sample JSON data as shown below.

### HTML

```
/// <reference path="jquery.d.ts" />
```

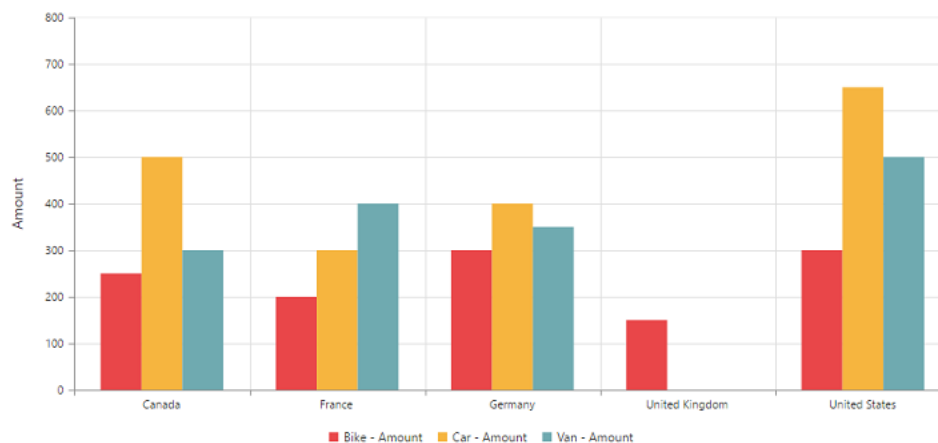
```

/// <reference path="ej.web.all.d.ts" />
var pivot_dataset = [
{ Amount: 100, Country: "Canada", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Alberta" },
{ Amount: 200, Country: "Canada", Date: "FY 2006", Product: "Van",
Quantity: 3, State: "British Columbia" },
{ Amount: 300, Country: "Canada", Date: "FY 2007", Product: "Car",
Quantity: 4, State: "Brunswick" },
{ Amount: 150, Country: "Canada", Date: "FY 2008", Product: "Bike",
Quantity: 3, State: "Manitoba" },
{ Amount: 200, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 4, State: "Ontario" },
{ Amount: 100, Country: "Canada", Date: "FY 2007", Product: "Van",
Quantity: 1, State: "Quebec" },
{ Amount: 200, Country: "France", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Charente-Maritime" },
{ Amount: 250, Country: "France", Date: "FY 2006", Product: "Van",
Quantity: 4, State: "Essonne" },
{ Amount: 300, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 3, State: "Garonne (Haute)" },
{ Amount: 150, Country: "France", Date: "FY 2008", Product: "Van",
Quantity: 2, State: "Gers" },
{ Amount: 200, Country: "Germany", Date: "FY 2006", Product: "Van",
Quantity: 3, State: "Bayern" },
{ Amount: 250, Country: "Germany", Date: "FY 2007", Product: "Car",
Quantity: 3, State: "Brandenburg" },
{ Amount: 150, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 4, State: "Hamburg" },
{ Amount: 200, Country: "Germany", Date: "FY 2008", Product: "Bike",
Quantity: 4, State: "Hessen" },
{ Amount: 150, Country: "Germany", Date: "FY 2007", Product: "Van",
Quantity: 3, State: "Nordrhein-Westfalen" },
{ Amount: 100, Country: "Germany", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Saarland" },
{ Amount: 150, Country: "United Kingdom", Date: "FY 2008", Product:
"Bike", Quantity: 5, State: "England" },
{ Amount: 250, Country: "United States", Date: "FY 2007", Product: "Car",
Quantity: 4, State: "Alabama" },
{ Amount: 200, Country: "United States", Date: "FY 2005", Product: "Van",
Quantity: 4, State: "California" },
{ Amount: 100, Country: "United States", Date: "FY 2006", Product:
"Bike", Quantity: 2, State: "Colorado" },
{ Amount: 150, Country: "United States", Date: "FY 2008", Product: "Car",
Quantity: 3, State: "New Mexico" },
{ Amount: 200, Country: "United States", Date: "FY 2005", Product:
"Bike", Quantity: 4, State: "New York" },
{ Amount: 250, Country: "United States", Date: "FY 2008", Product: "Car",
Quantity: 3, State: "North Carolina" },
{ Amount: 300, Country: "United States", Date: "FY 2007", Product: "Van",
Quantity: 4, State: "South Carolina" }
];
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
dataSource: {
data: pivot_dataset,
rows: [
{

```

```
fieldName: "Country",
fieldCaption: "Country"
},
{
  fieldName: "State",
  fieldCaption: "State"
},
{
  fieldName: "Date",
  fieldCaption: "Date"
}
],
columns: [
  {
    fieldName: "Product",
    fieldCaption: "Product"
  }
],
values: [
  {
    fieldName: "Amount",
    fieldCaption: "Amount"
  }
],
filters: []
},
isResponsive: true,
commonSeriesOptions: {
  type: ej.PivotChart.ChartTypes.Column
},
size: { height: "460px", width: "950px" },
primaryYAxis: { title: { text: "Amount" } },
legend: { visible: true }
});
```

The above code will generate a simple PivotChart with sales amount over products in different regions.



## OLAP

This section covers the information that you need to know to populate a simple PivotChart with OLAP data source.

### Control Initialization

Add a `div` container to render the PivotChart.

#### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="PivotChart1" style="min-height: 275px; min-width: 525px; height: 460px; width: 100%"></div>
</body>
</html>
```

Initialize the PivotChart in ts file by using the `ej.PivotChart` method.

#### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), { });
});
```

### Populate PivotChart with data

Let us now see how to populate the PivotChart control using a sample JSON data as shown below.

#### HTML

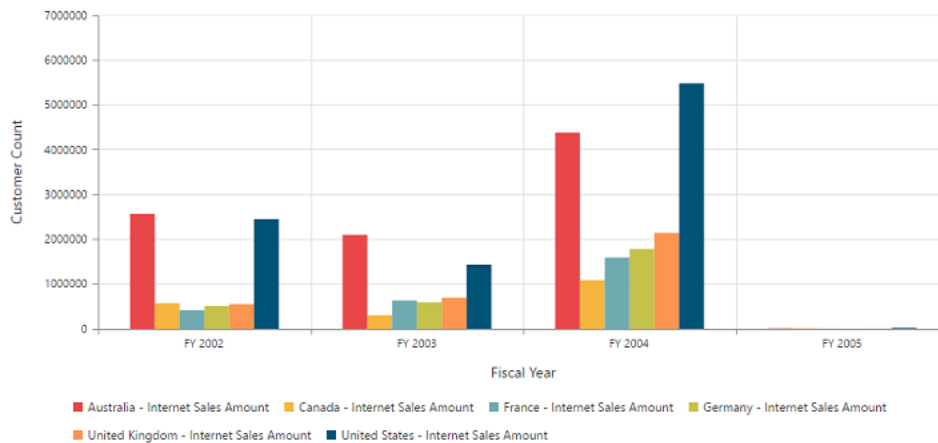
```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
dataSource: {
data: "http://bi.syncfusion.com/olap/msmdpump.dll",
catalog: "Adventure Works DW 2008 SE",
cube: "Adventure Works",
rows: [
{
fieldName: "[Date].[Fiscal]"
}
],
columns: [
{
fieldName: "[Customer].[Customer Geography]"
}
],
values: [
{
measures: [
{
fieldName: "[Measures].[Internet Sales Amount]"
}
]
}
]
});
```

```

],
axis: "columns"
}
],
filters: []
},
isResponsive: true,
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.Column
},
size: { height: "460px", width: "950px" },
primaryXAxis: { title: { text: "Date - Fiscal" }, labelRotation: 0 },
primaryYAxis: { title: { text: "Internet Sales Amount" } },
legend: { visible: true, rowCount: 2 }
});
});

```

The above code will generate a simple PivotChart with internet sales amount over a period of fiscal years across different customer geographic locations.



## Dimensions

### Set size in percentage

You can customize the pivot chart dimension by setting the width and height of the control in percentage.

### TS

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
size: {
height: "80%",
width: "80%"
},
});
});

```

### HTML



```
<style>
#PivotChart1 {
width:100%;
height:450px;
}
</style>
```

Set size in pixels

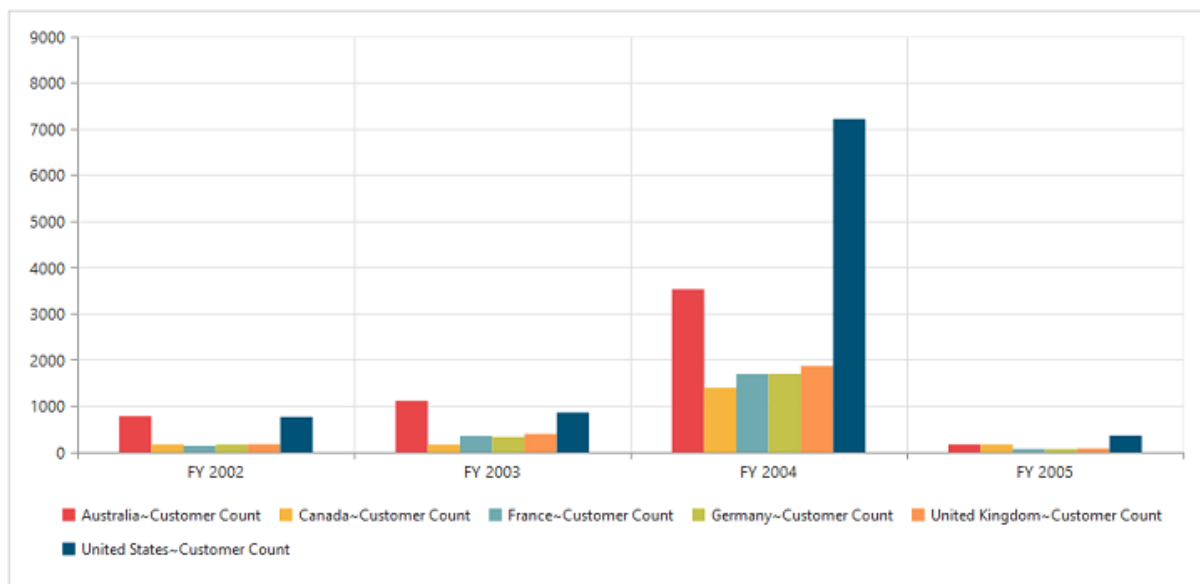
You can customize the pivot chart dimension by setting the width and height of the control in pixels.

## TS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
size: {
height: "460px",
width: "950px"
},
});
});
```

## HTML

```
<style>
#PivotChart1 {
width:950px;
height:460px;
}
</style>
```



## Responsive

The pivot chart control supports responsive rendering based on the target device (desktop and tablet) resolution. It supports resolution upto 1024x600. You can enable responsiveness in the pivot chart by setting the `IsResponsive` property to true.

## TS

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        isResponsive: true,
        size: {
            height: "460px",
            width: "950px"
        },
    });
});

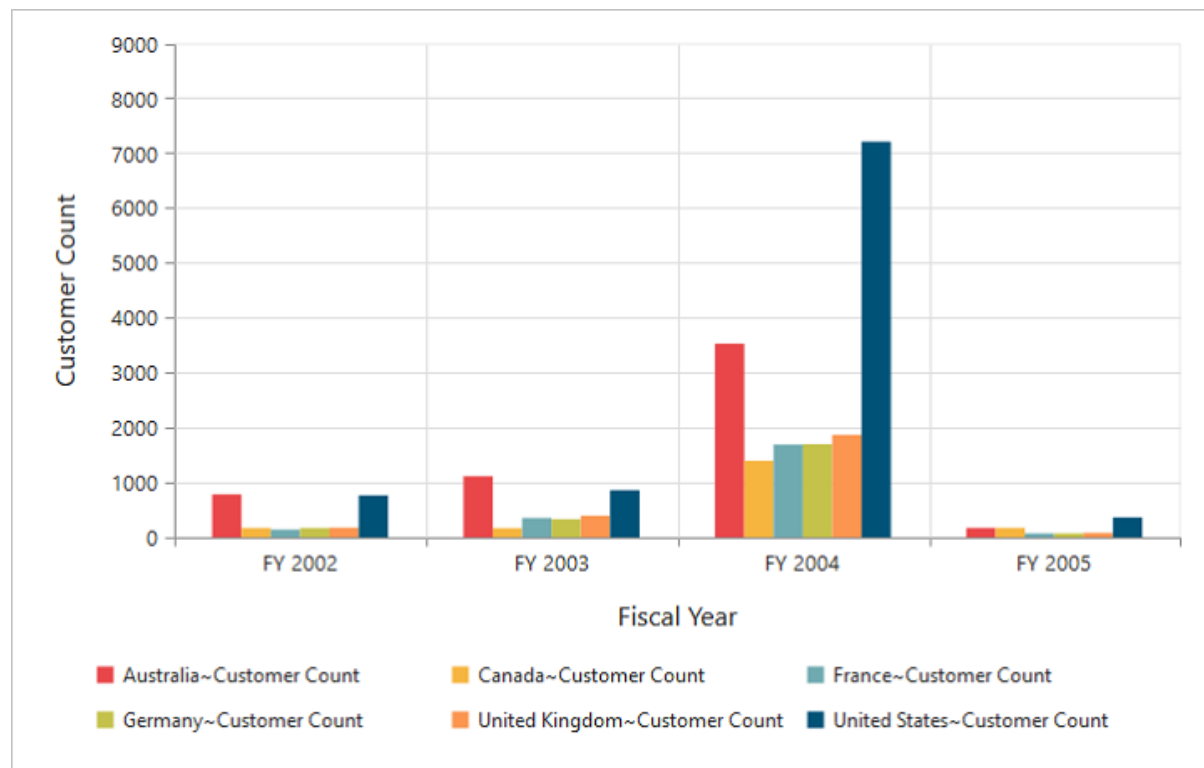
```

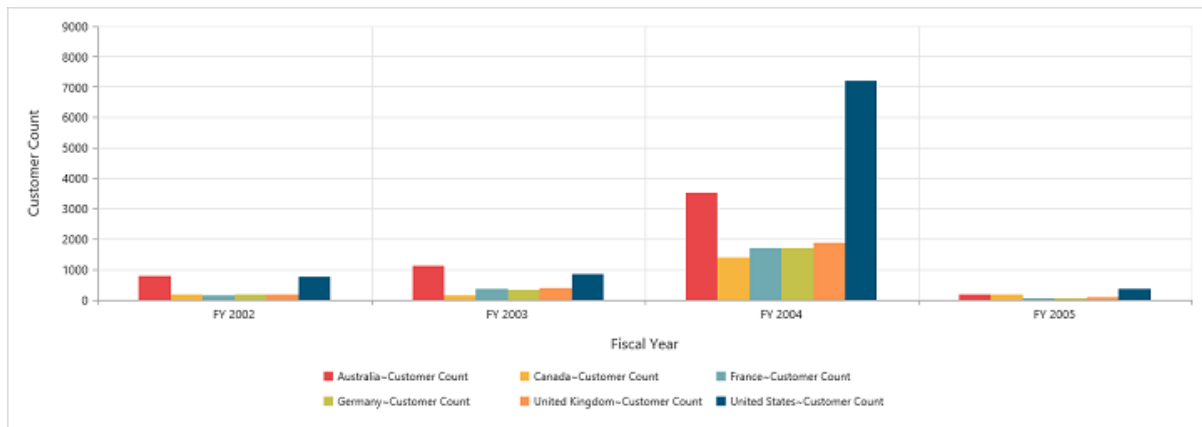
## HTML

```

<style>
#PivotChart1 {
    min-height: 275px;
    min-width: 525px;
    height: 460px;
    width: 100%;
}
</style>

```



*Normal View**ResponsiveView*

## Chart types

The Essential **PivotChart Typescript** supports 17 types of charts:

- Column
- Stacking column
- Bar
- Stacking bar
- Pie
- Pyramid
- Funnel
- Line
- Step line
- Spline
- Area
- Step area
- Spline area
- Stacking area
- Doughnut
- Scatter
- Bubble

## Column chart

The **column chart** is the most commonly used chart type. This uses vertical bars (called columns) to display different values of one or more items. Points from adjacent series are drawn as bars to compare frequency, count, total, or average of the data in different categories. Column chart is ideal to show the variations in the value of an item over a period of time.

**HTML**

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        commonSeriesOptions: {

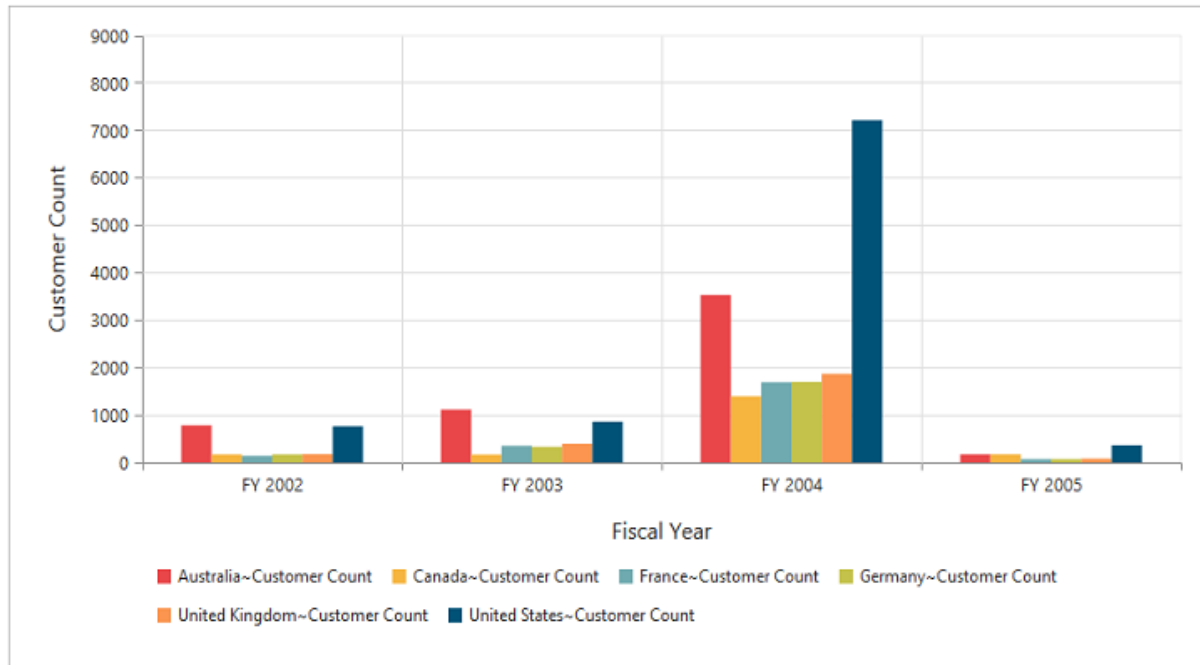
```

```

type: ej.PivotChart.ChartTypes.Column
},
});
});

```

The following screenshot displays **column chart**:



### Stacking column chart

The **stacking column** chart is similar to column charts except for the Y-values. The Y-values stack on top of each other in a specified series order. This helps to visualize the relationship of parts to the whole chart across various categories.

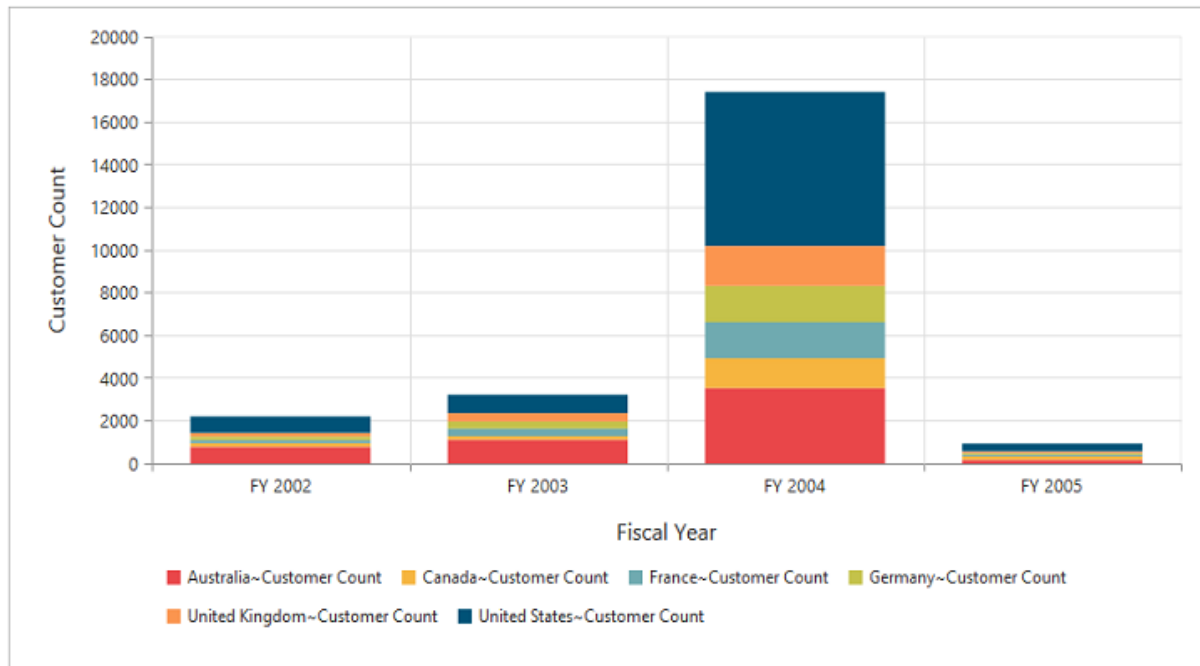
### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.StackingColumn
},
});
});
});

```

The following screenshot displays **stacking column chart**:



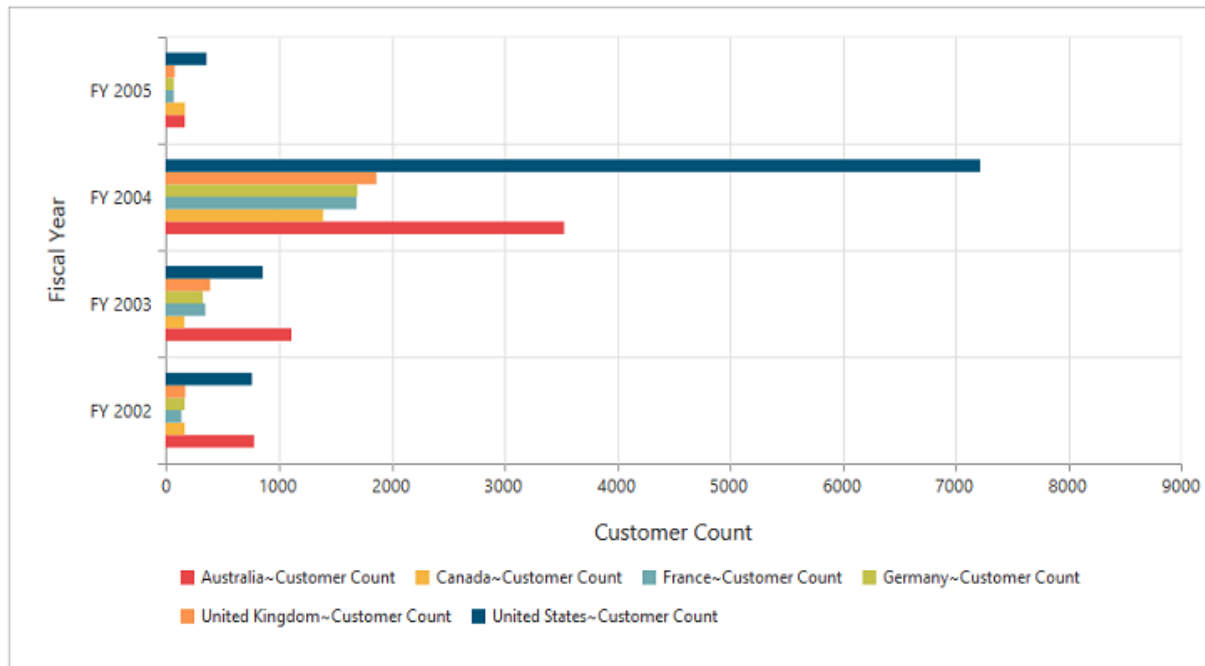
### Bar chart

The **bar chart** displays horizontal bars for each point in the adjacent series. Bar charts are used to compare values across various categories for displaying the variations in the value of an item over a period of time or comparing the values of several items at a single point of time.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        commonSeriesOptions: {
            type: ej.PivotChart.ChartTypes.Bar
        },
    });
});
```

The following screenshot displays **bar chart**:



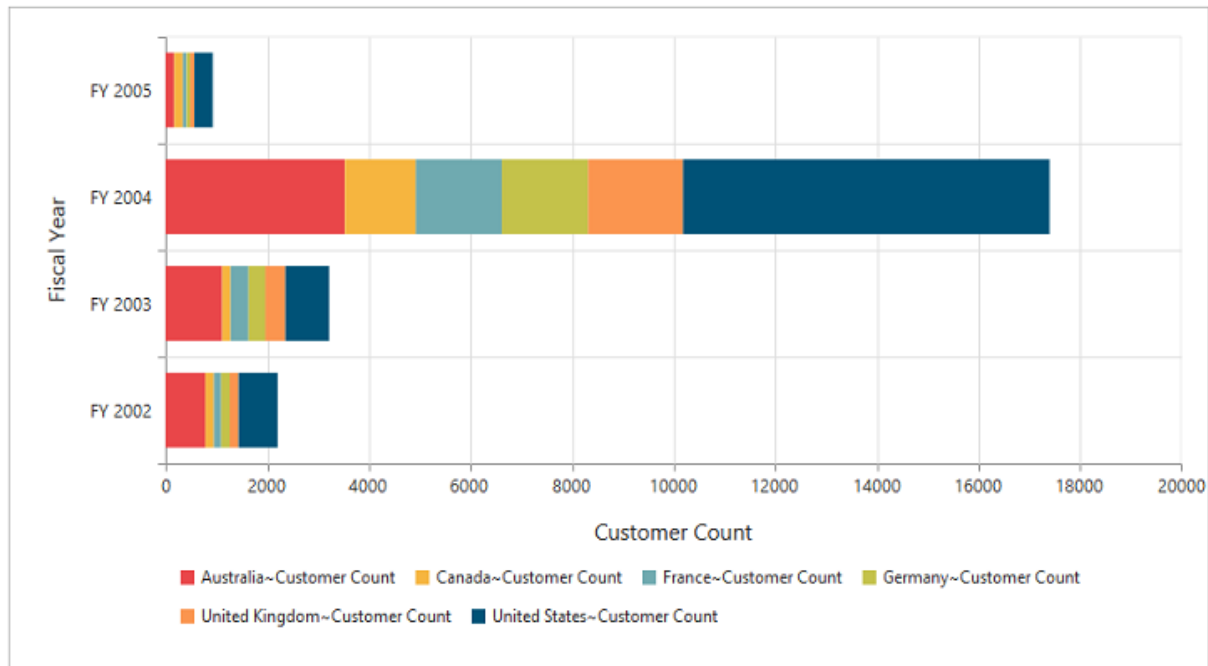
### Stacking bar chart

The **stacking bar chart** is a regular **bar** chart with the X-values stacked on top of each other in the specified series order.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        commonSeriesOptions: {
            type: ej.PivotChart.ChartTypes.StackingBar
        },
    });
});
```

The following screenshot displays **stacking bar chart**:



### Pie chart

The **pie chart** is used to summarize a set of categorical data or display different values of a given variable (e.g., percentage distribution). This type of chart is in a circular form that is divided into several segments. Each segment represents a particular category.

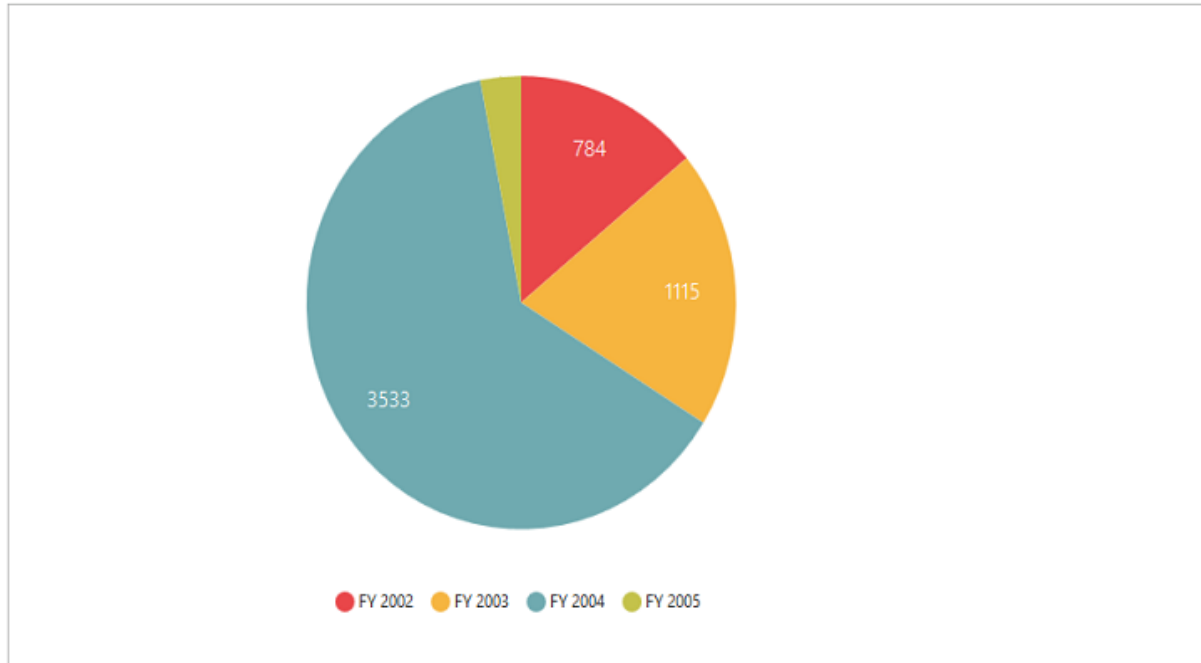
### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.Pie
},
});
});

```

The following screenshot displays **pie chart**:



### Pyramid chart

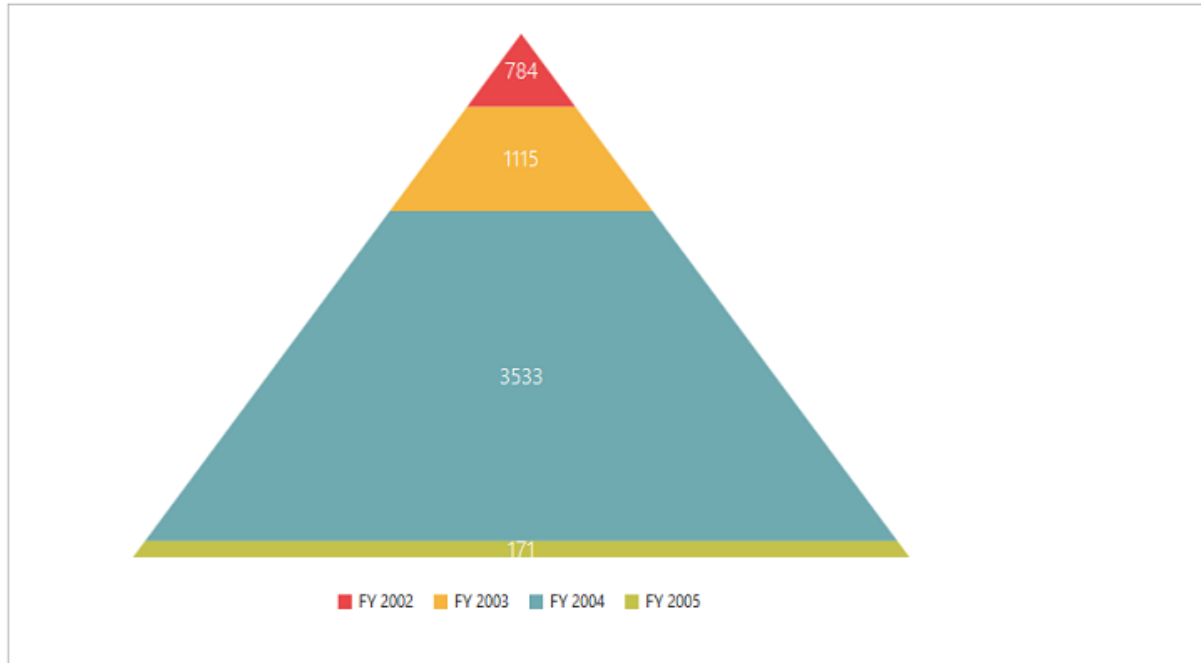
The **pyramid chart** displays data in the form of a triangle. You can visualize data in a hierarchical structure without any axes.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
  var sample = new ej.PivotChart($("#PivotChart1"), {
    commonSeriesOptions: {
      type: ej.PivotChart.ChartTypes.Pyramid
    },
  });
});
```

The following screenshot displays **pyramid chart**:





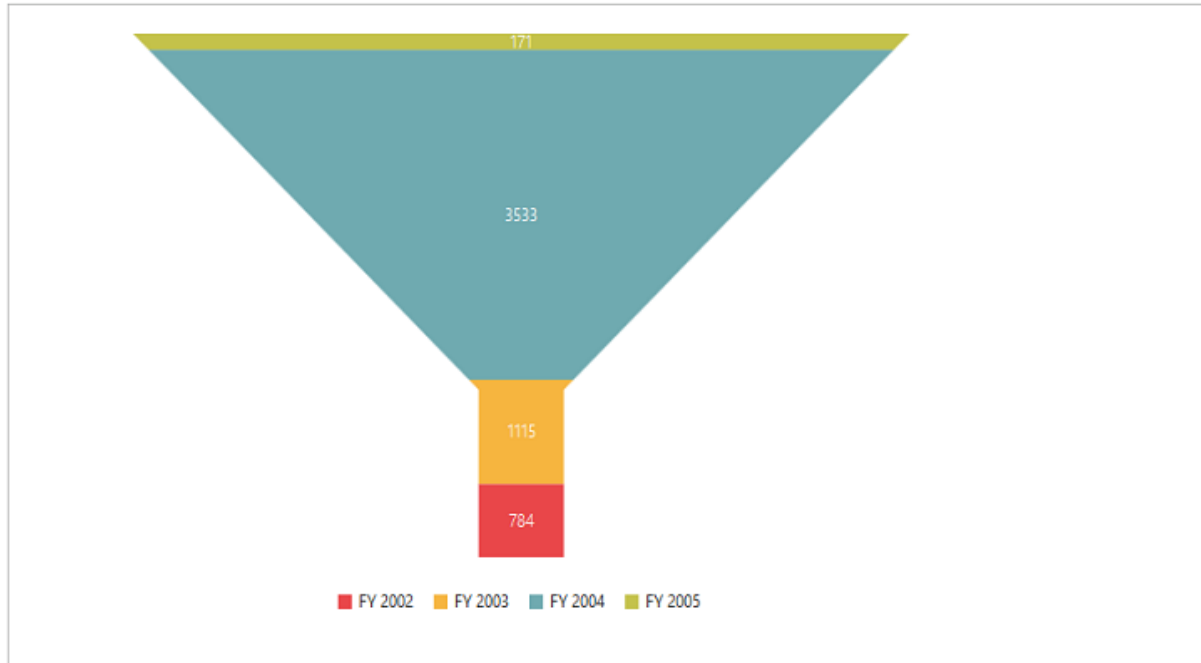
### Funnel chart

The **funnel chart** displays data in the form of an inverted triangle. You can visualize data in a hierarchical structure without any axes.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.Funnel
},
});
});
```

The following screenshot displays **funnel chart**:



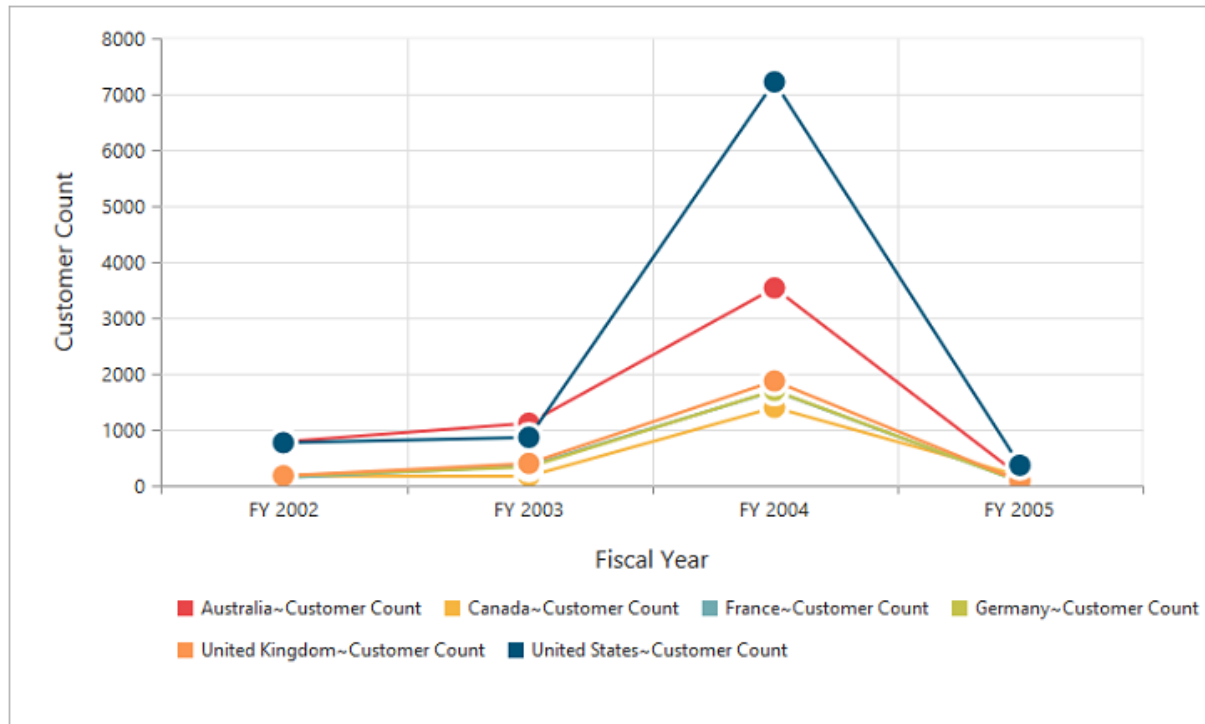
### Line chart

The **line chart** joins the data points on a plot using straight lines that show the trends in data at equal intervals.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        commonSeriesOptions: {
            type: ej.PivotChart.ChartTypes.Line
        },
    });
});
```

The following screenshot displays the **line chart**:



### Step line chart

The **step line chart** uses horizontal and vertical lines to connect data points resulting in a step like progression.

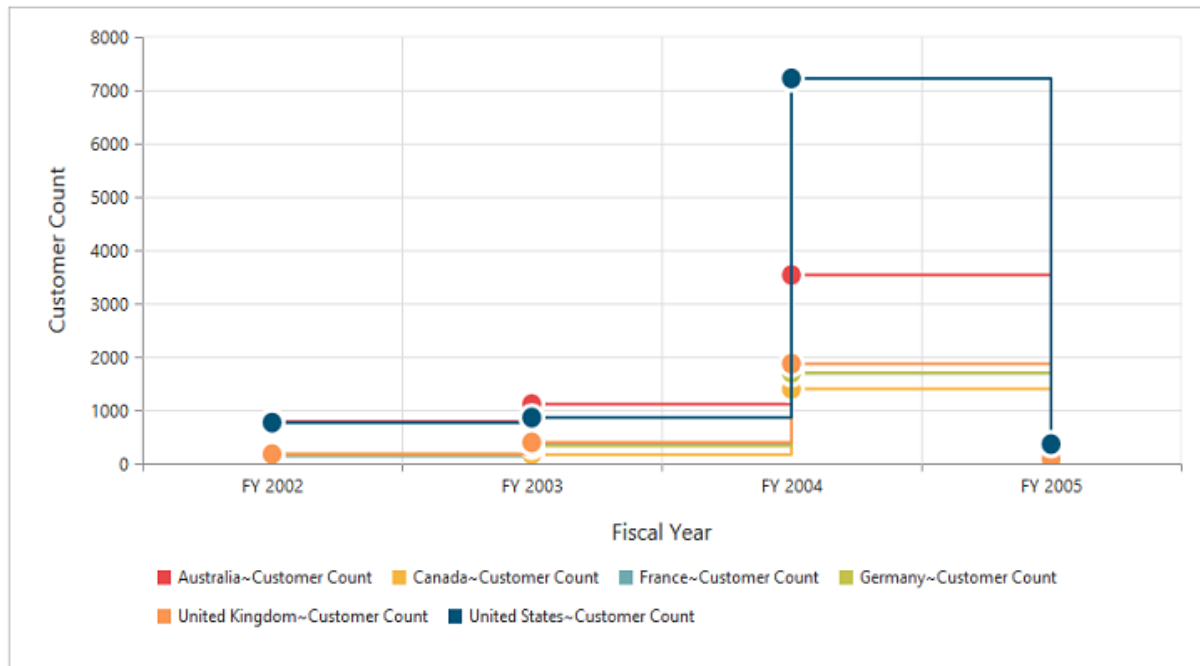
### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        commonSeriesOptions: {
            type: ej.PivotChart.ChartTypes.StepLine
        },
    });
});

```

The following screenshot displays **step line chart**:



### Spline chart

The **spline chart** is similar to the line chart except that it connects different data points with curve lines instead of straight lines.

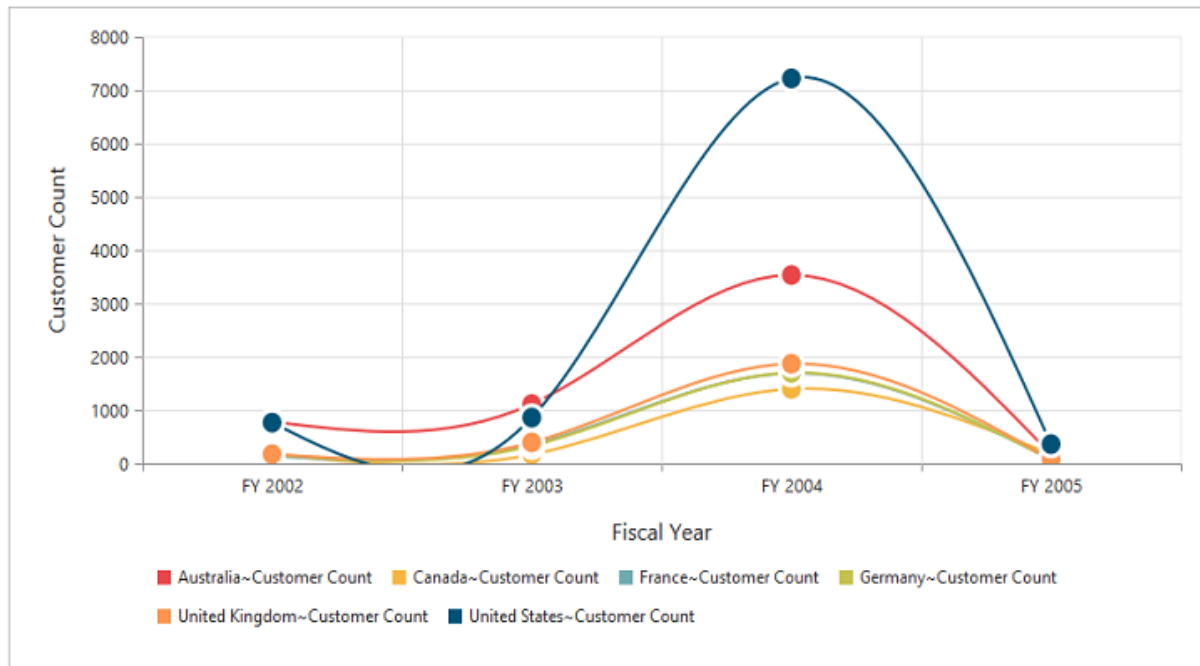
### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.Spline
},
});
});

```

The following screenshot displays the **spline chart**:



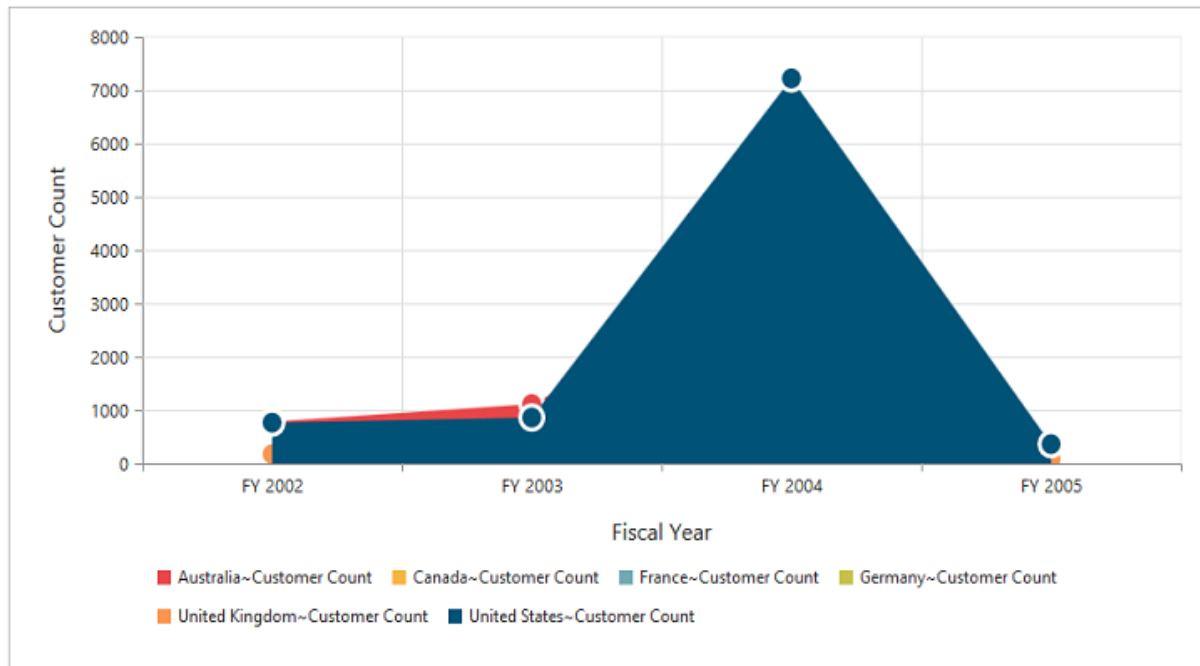
### Area chart

The **area chart** emphasizes the degree of change of values over a period of time. Instead of rendering data as discrete bars or columns, the area chart renders the continuous ebb-and-flow pattern as defined against the Y-axis.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        commonSeriesOptions: {
            type: ej.PivotChart.ChartTypes.Area
        },
    });
});
```

The following screenshot displays **area chart**:



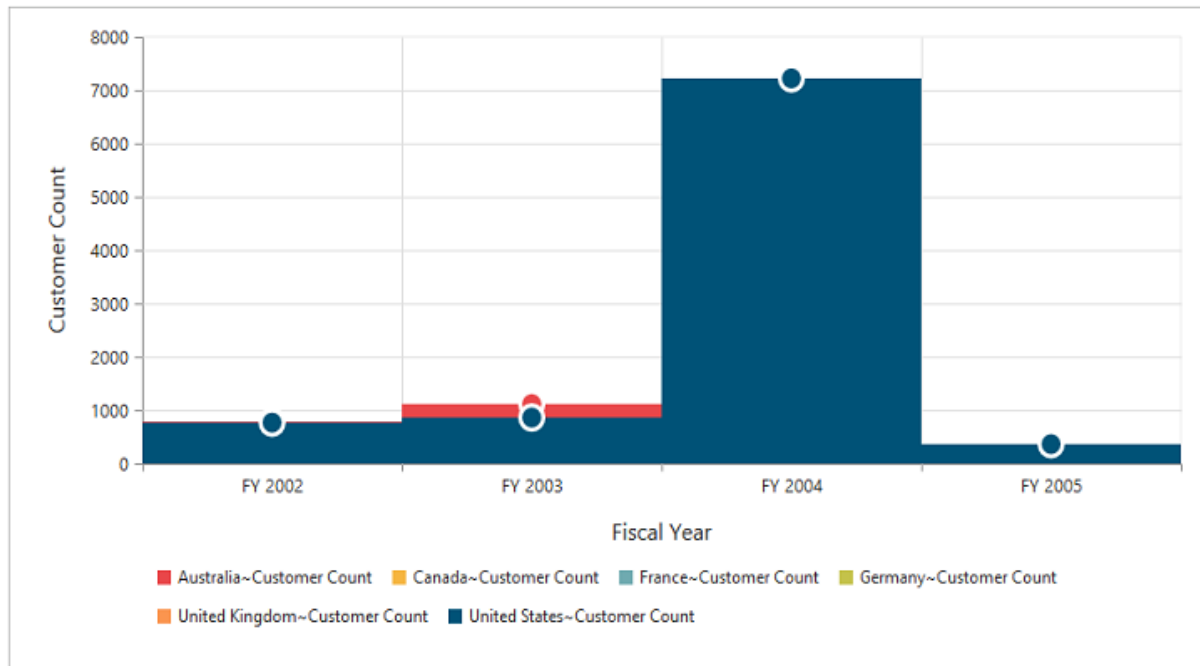
### Step area chart

The **step area** chart is similar to the regular area chart except for a straight line tracing the shortest path between the data points. The values are connected by continuous vertical and horizontal lines to form a step like progression.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        commonSeriesOptions: {
            type: ej.PivotChart.ChartTypes.StepArea
        },
    });
});
```

The following screenshot displays **step area chart**:



### Spline area chart

The **spline area** chart is similar to the area chart, but differs by connecting data points in a series. This connects each series of points by a smooth **spline curve**.

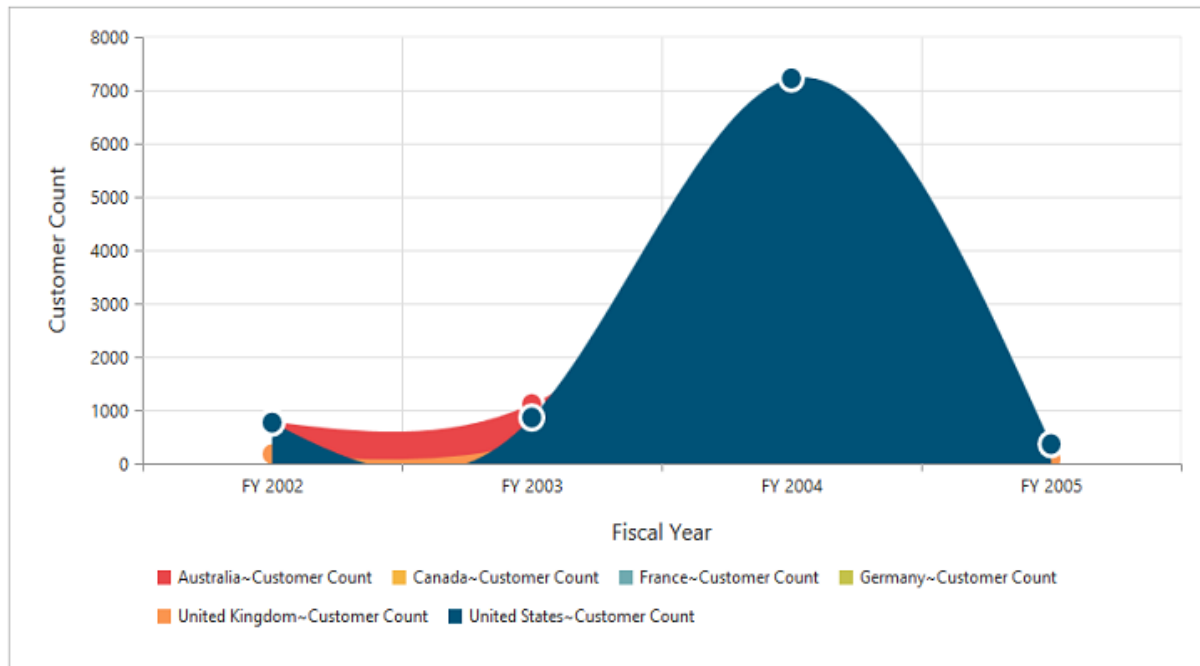
### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.SplineArea
},
});
});

```

The following screenshot displays **spline area chart**:



### Stacking area chart

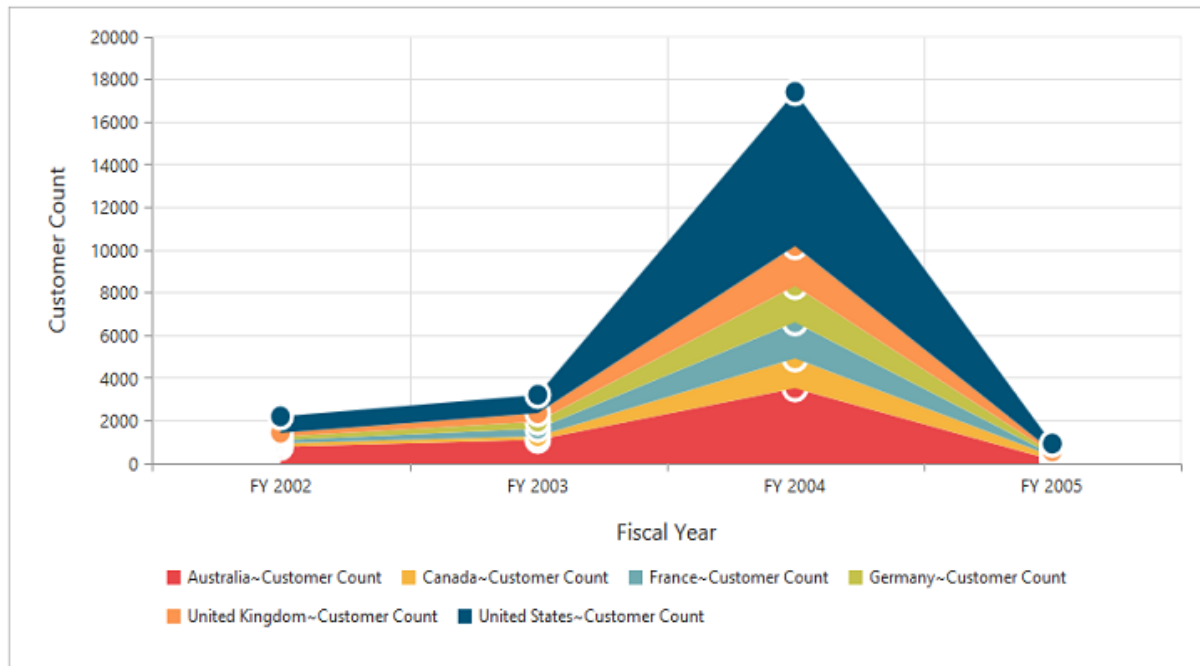
The **stacking area** chart is similar to the regular area chart except for the Y-values. These Y-values stack on top of each other in the specified series order. This helps to visualize the relationship of parts to the whole chart across various categories.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        commonSeriesOptions: {
            type: ej.PivotChart.ChartTypes.StackingArea
        },
    });
});
```

The following screenshot displays **stacking area chart**:





### Doughnut chart

The **doughnut chart** is a doughnut like structure used to summarize a set of categorical data that is divided into several segments. Each segment represents a particular category.

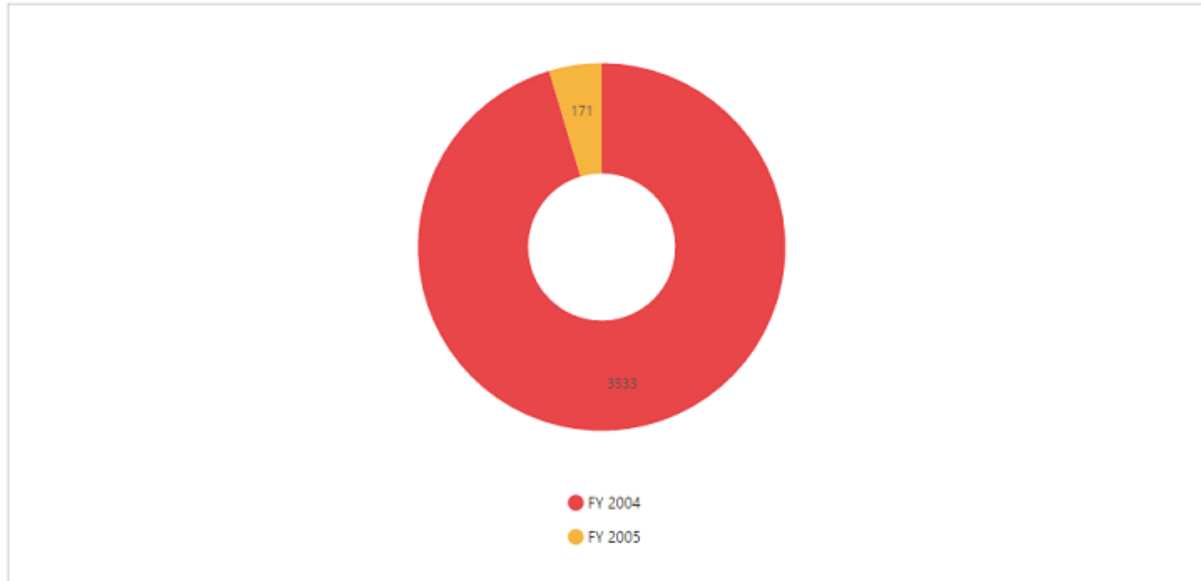
### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.Doughnut
},
});
});

```

The following screenshot displays **doughnut chart**:



### Scatter chart

The **scatter chart** displays data as a collection of points corresponding to associated values.

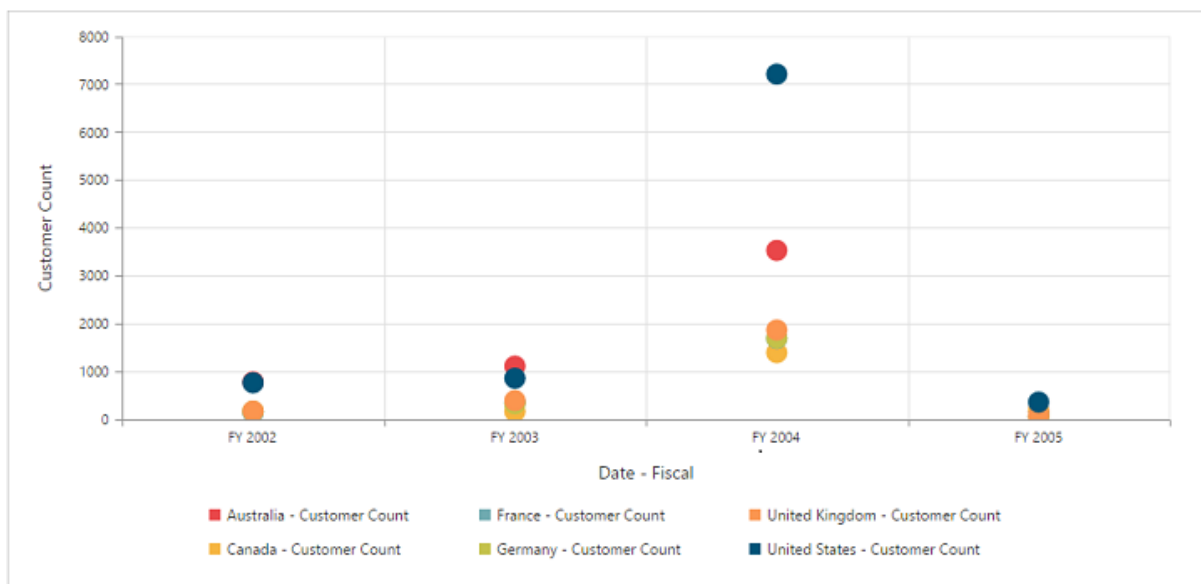
### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.Scatter
},
});
});

```

The following screenshot displays **scatter chart**:



### Bubble chart

The **bubble chart** displays data as a collection of bubbles.

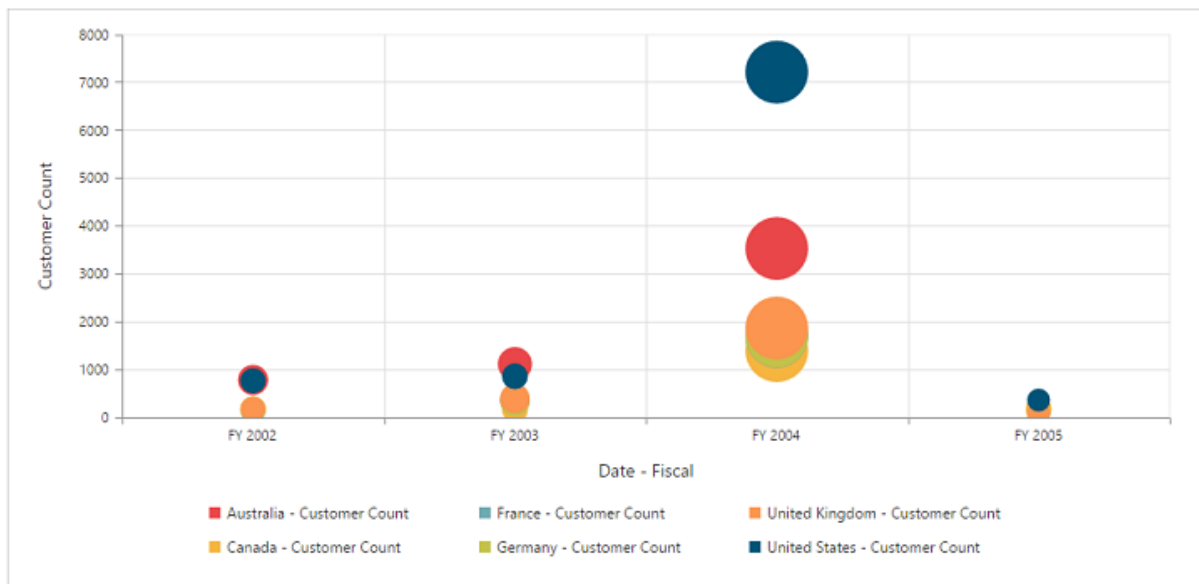
#### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.Bubble
},
});
});

```

The following screenshot displays **bubble chart**:



### Combination chart

The **combination chart** combines two or more series types in a single chart, but there are some limitations in the combination chart. They are:

1. The combination chart cannot combine column and bar series.
2. The pie chart cannot be used with other series types.

#### HTML

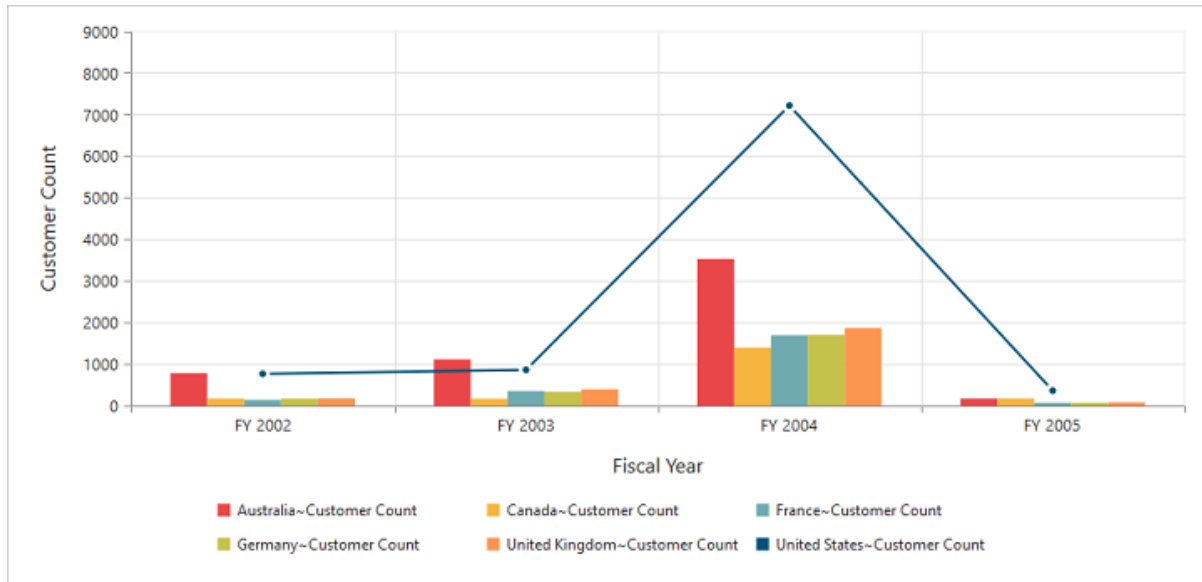
```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.Column
},
load: function (args) {
this.model.seriesRendering = function (event) {

```

```
this.model.series[5].type = ej.PivotChart.ChartTypes.Line;  
this.model.series[5].marker.visible = true;  
};  
}  
});  
});
```

The following screenshot displays **combination chart**:

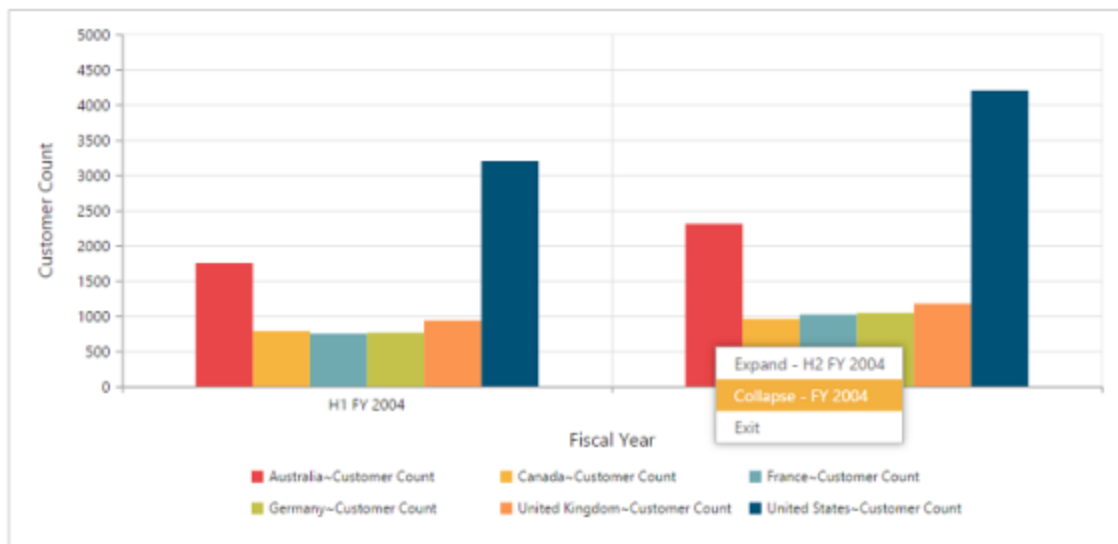
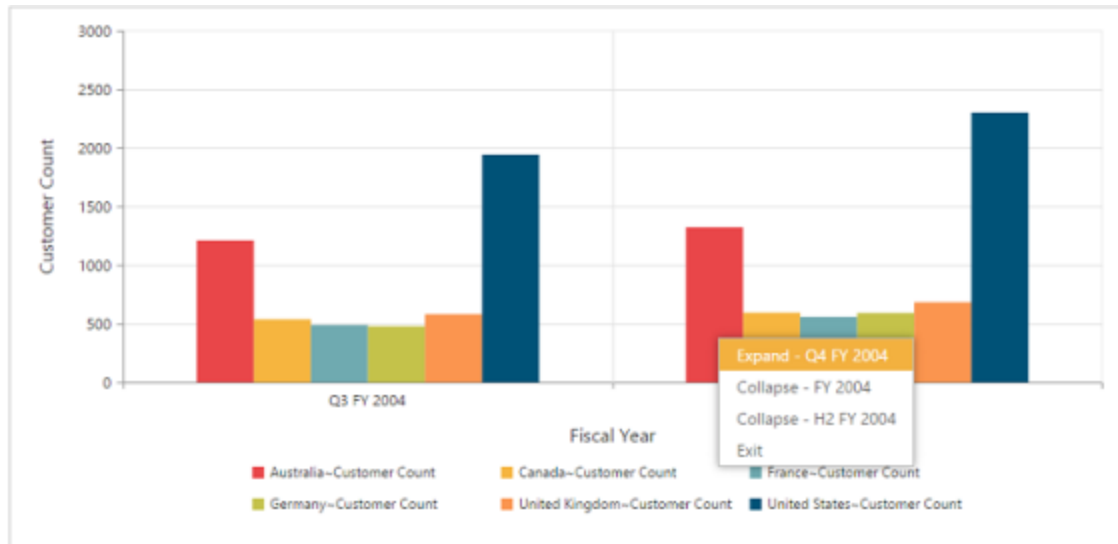


### Drill operation

This is a basic feature of the pivot chart through which the amount of information can be limited for a better view. It allows you to drill down to access the detailed level of data or drill up to see the summarized data by using the context menu present in the pivot chart.

Drill up, also called roll up, navigates from the inner most level (having detailed member information) to any of the outer levels.

Drill down, also called roll down, is the reverse of drill up. It navigates from any of the outer levels to the inner most level.



The drillSuccess event will be triggered when you right-click the pivot chart and select any option available from the context menu to perform drill up or drill down operation.

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        drillSuccess: function (args) {
            alert("Drill Success");
        }
    });
});

```

### Legend

#### Legend visibility

You can enable or disable the legend by using the `visible` property in the `legend` object.

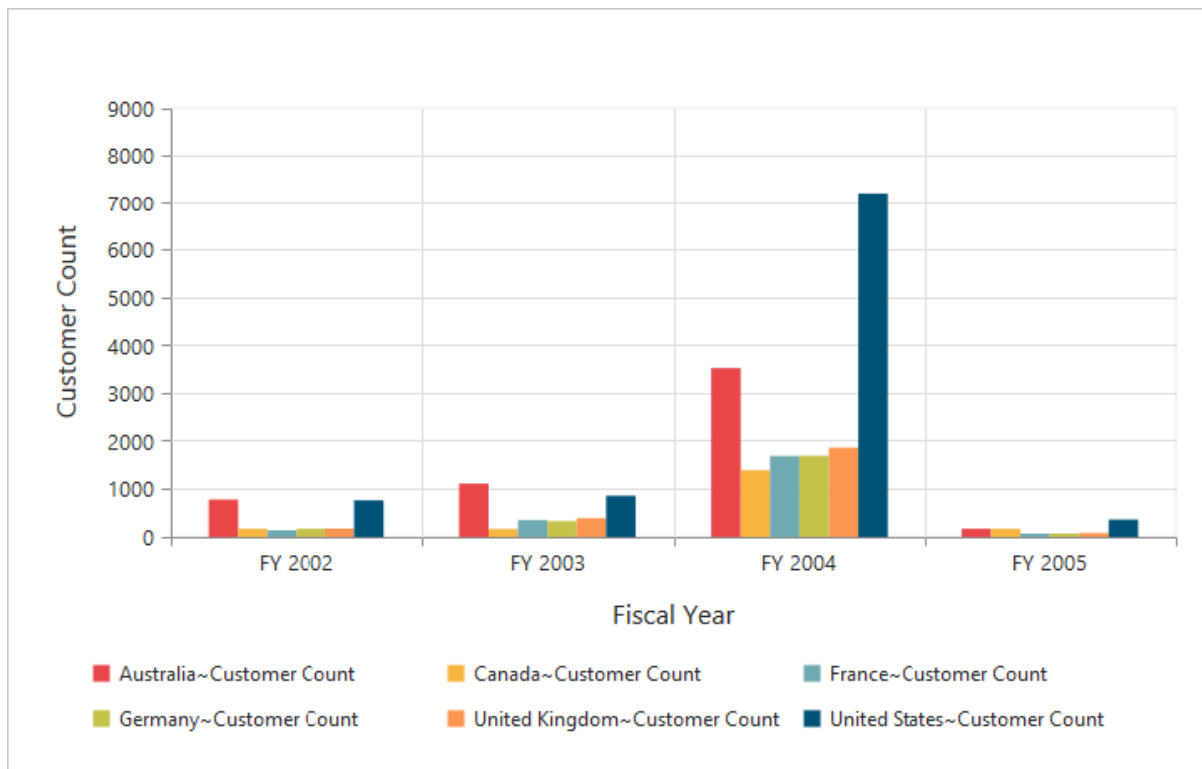
**Note:** By default, the legend is visible in the pivot chart.

### HTML

```

// <reference path="jquery.d.ts" />
// <reference path="ej.web.all.d.ts" />
$(function () {
  var sample = new ej.PivotChart($("#PivotChart1"), {
    legend: {
      visible: true
    },
  });
});

```



### Legend shape

You can customize the legend **shape** in the pivot chart control. The default value of legend shape is rectangle. Following are the legend shapes that are supported:

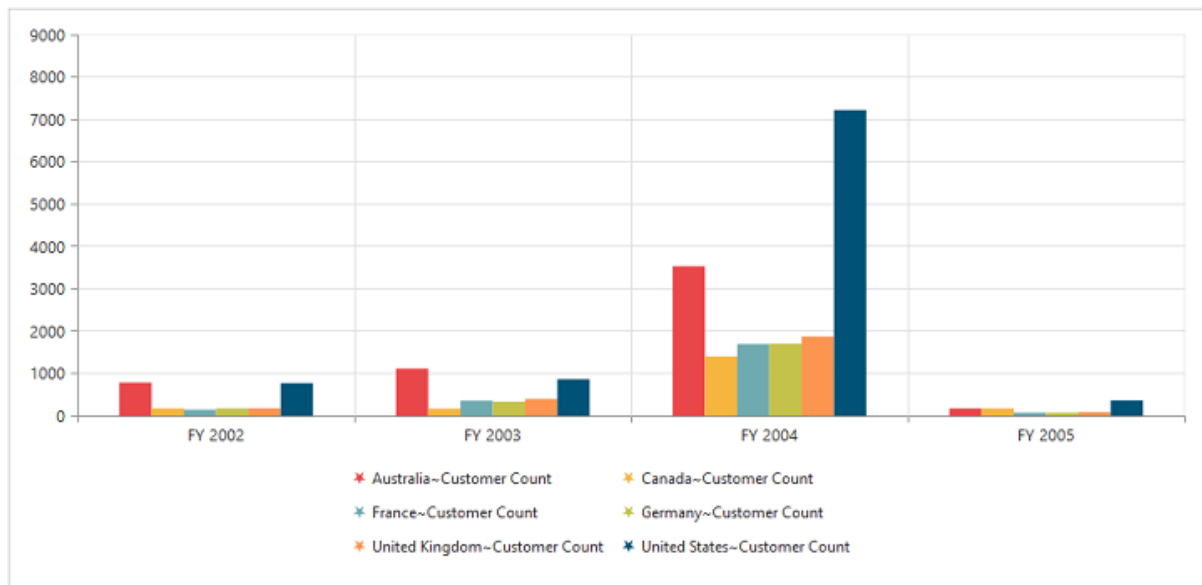
- Rectangle
- Circle
- Cross
- Diamond
- Pentagon
- Hexagon
- Star
- Ellipse
- Triangle etc.

**HTML**

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
  legend: {
    visible: true,
    shape: "star"
  },
});
});

```

**Legend position**

By using the `position` property, you can place the legend at top, bottom, left, or right of the pivot chart.

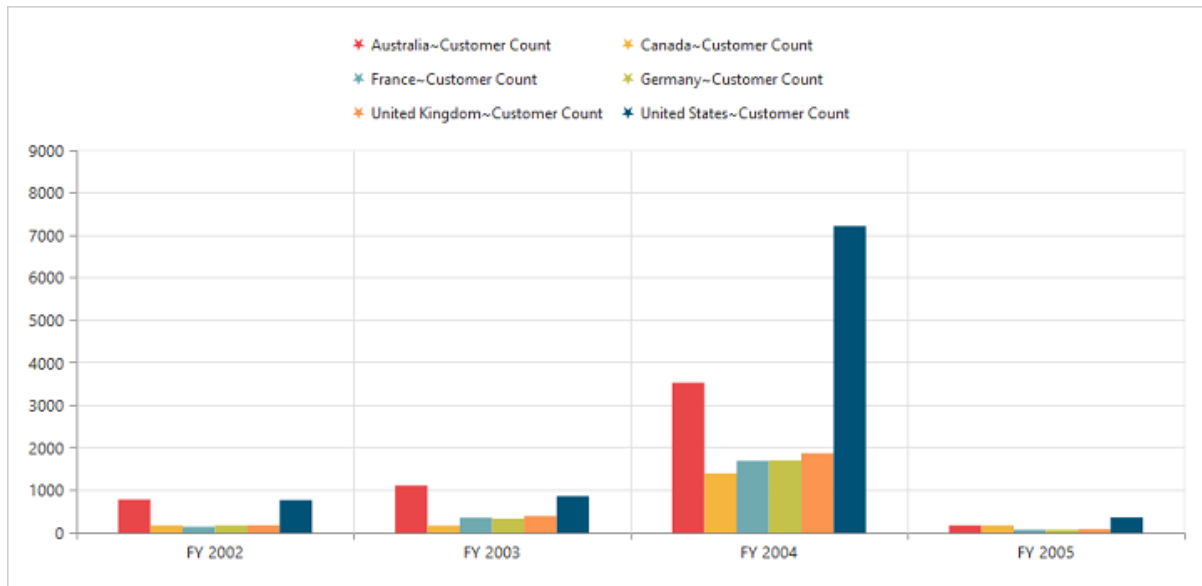
**Note:** The default value of legend position is bottom in the pivot chart.

**HTML**

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
  legend: {
    visible: true,
    position: "top"
  },
});
});

```



### Legend title

To add the legend title, you should specify the title text in the `title.text` property.

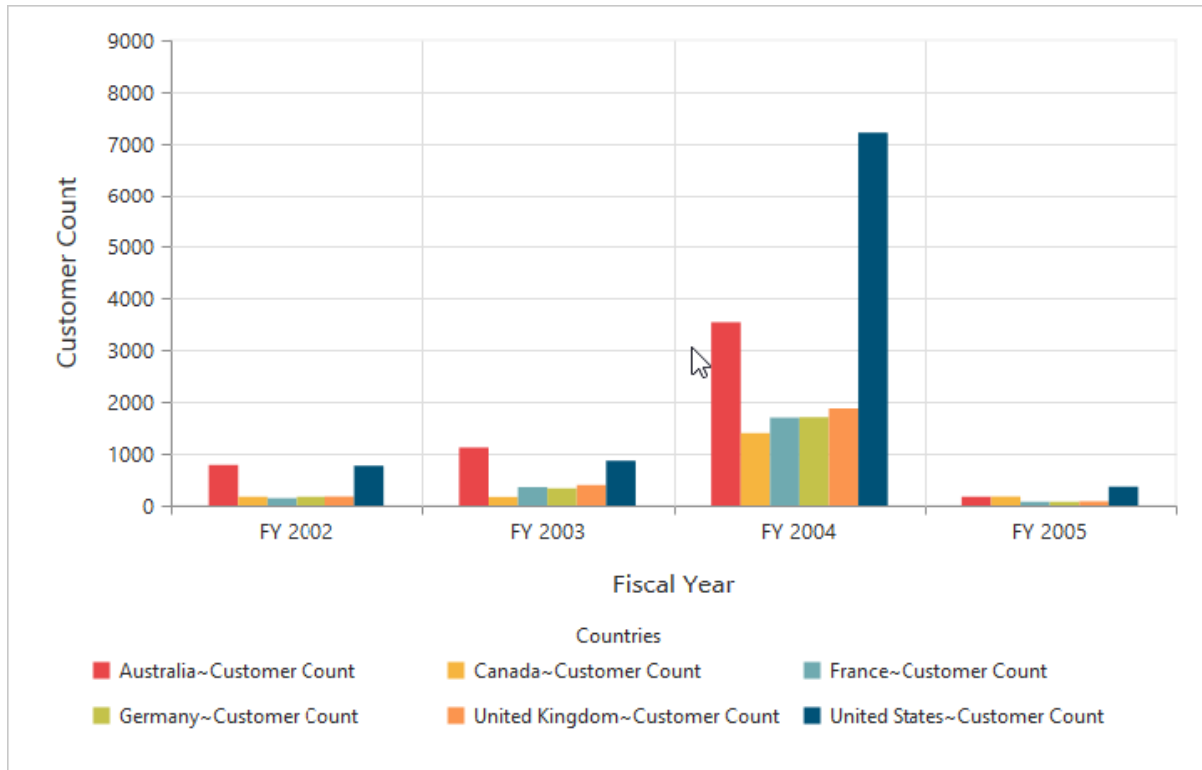
### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
  var sample = new ej.PivotChart($("#PivotChart1"), {
    legend: {
      visible: true,
      title: {
        text : "Countries"
      }
    },
  });
});

```





### Legend alignment

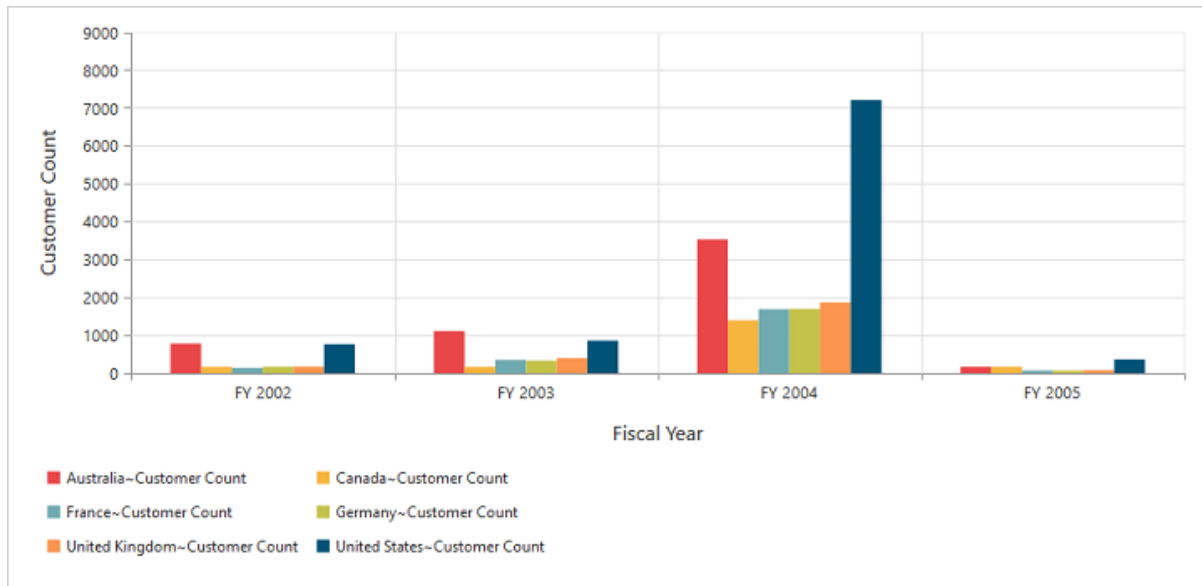
You can align the legend to center, far, and near based on its position in the chart area by using the `alignment` option.

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
  legend: {
    visible: true,
    alignment: "near"
  },
});
});

```



### Legend items - size and border

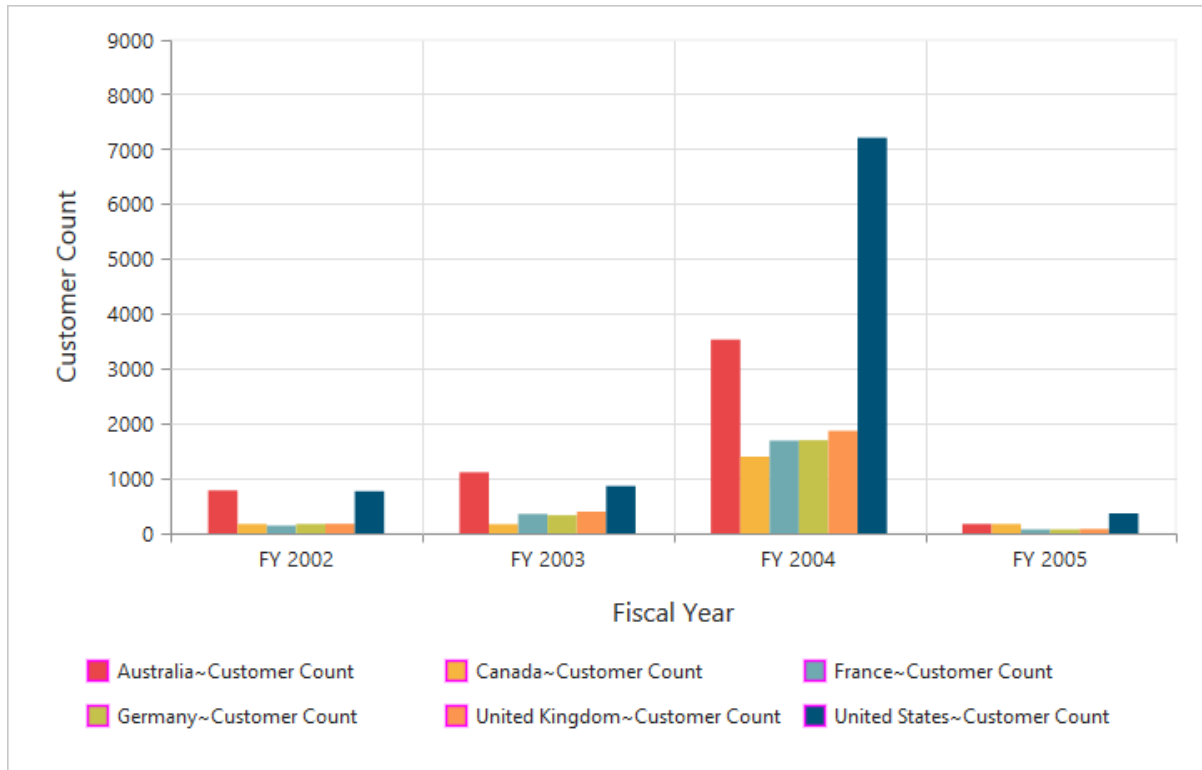
By using the legend `itemStyle.width`, `itemStyle.height`, and `itemStyle.border` properties, you can change the size and border of legend items.

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        legend: {
            visible: true,
            itemStyle: {
                border: {
                    width: 1.5,
                    color: "magenta"
                },
                width: 12,
                height: 12
            }
        },
    });
});

```



### Legend border

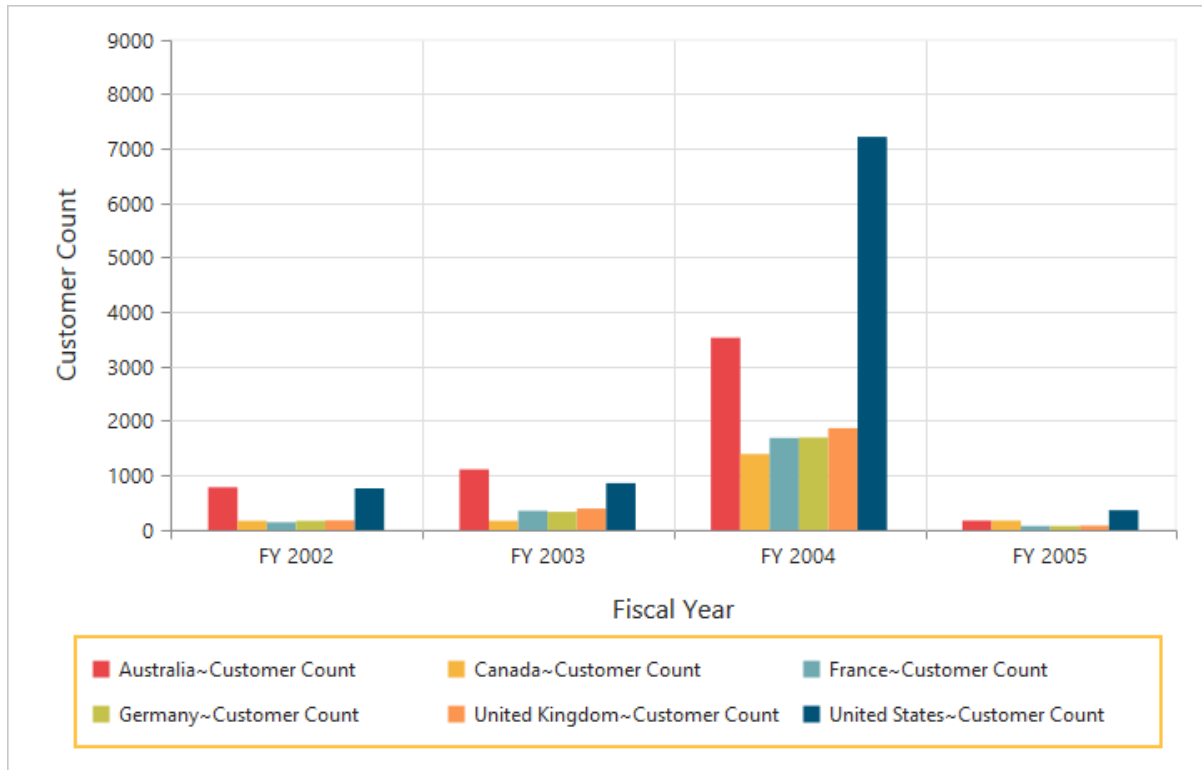
By using the **border** option in the legend, you can customize the border color and width.

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
  legend: {
    visible: true,
    border: {
      color: "orange",
      width: 5
    }
  },
});
});

```



### Legend text

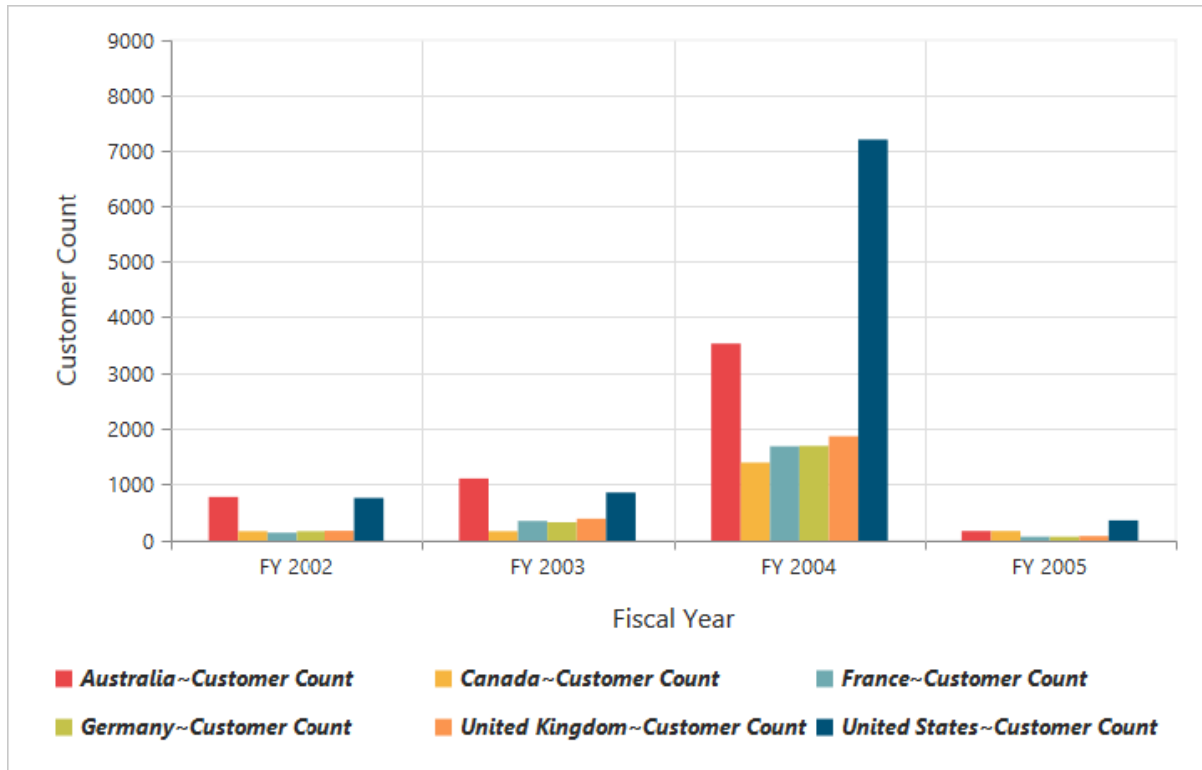
By using the **font** option, you can customize the font family, font style, font weight, and size of the legend text.

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
  legend: {
    visible: true,
    font: {
      size: "20px",
      color: "blue",
      fontWeight: "bold",
      fontFamily: "Segoe UI"
    }
  },
});
});

```



## Axes

### Label format

#### Format numeric labels

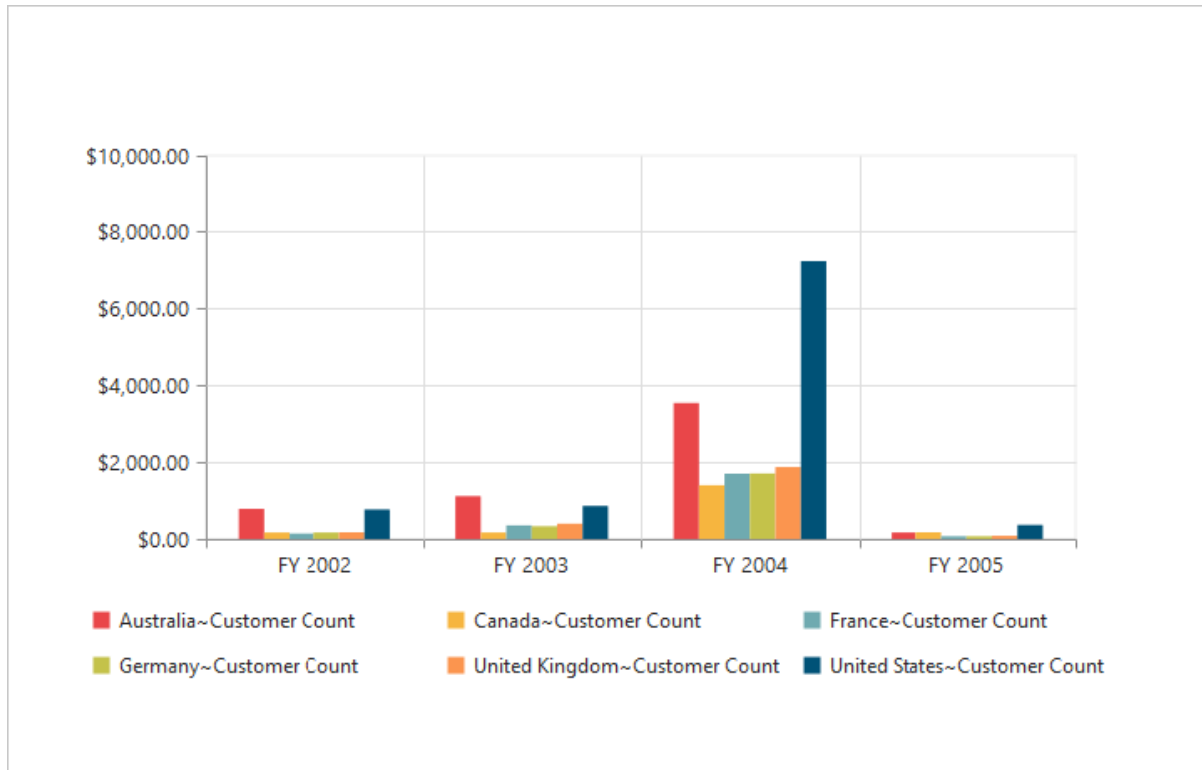
By using the `labelFormat` property, you can format the numeric labels. Numeric values can be formatted with `n` (number with decimal points), `c` (currency), and `p` (percentage) commands.

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
primaryYAxis: {
labelFormat: "c"
},
});
});

```



Following table describes the result when applying some commonly used label formats on numeric values:

| Label Value | Label Format Property Value | Result     | Description  |
|-------------|-----------------------------|------------|--|
| 1000        | n1                          | 1000.0     | The Number is rounded to 1 decimal place   |
| 1000        | n2                          | 1000.00    | The Number is rounded to 2 decimal place   |
| 1000        | n3                          | 1000.000   | The Number is rounded to 3 decimal place   |
| 0.01        | p1                          | 1.0%       | The Number is converted to percentage with 1 decimal place                         |
| 0.01        | p2                          | 1.00%      | The Number is converted to percentage with 2 decimal place                         |
| 0.01        | p3                          | 1.000%     | The Number is converted to percentage with 3 decimal place                         |
| 1000        | c1                          | \$1,000.0  | The Currency symbol is appended to number and number is rounded to 1 decimal place |
| 1000        | c2                          | \$1,000.00 | The Currency symbol is appended to number and number is rounded to 2 decimal place |

### Label format customization

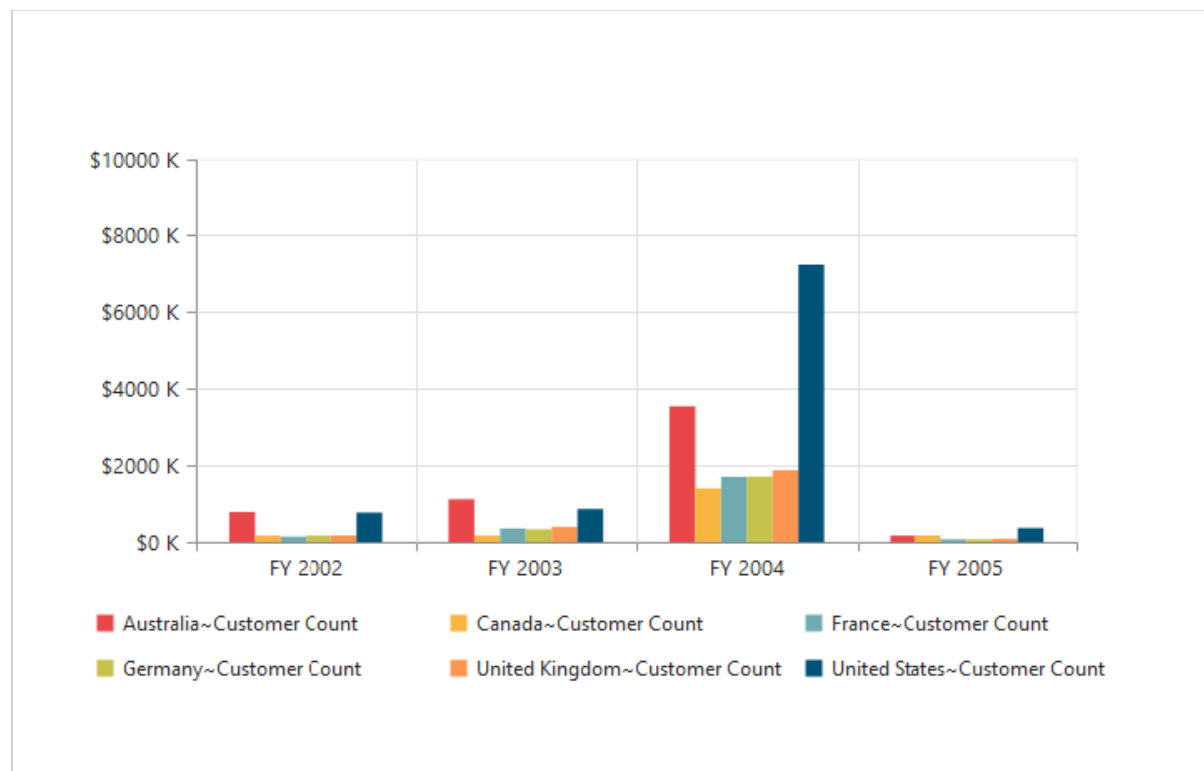
By using the `labelFormat` property of `primaryYAxis`, you can add the category labels with prefix and/or suffix.

#### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        primaryYAxis: {
            labelFormat: "${value}K"
        },
    });
});

```



### Common axis features

#### Axis visibility

Axis visibility can be set by using the `visible` property of the respective axis.

**Note:** By default, the value of `visible` property is true in the pivot chart.

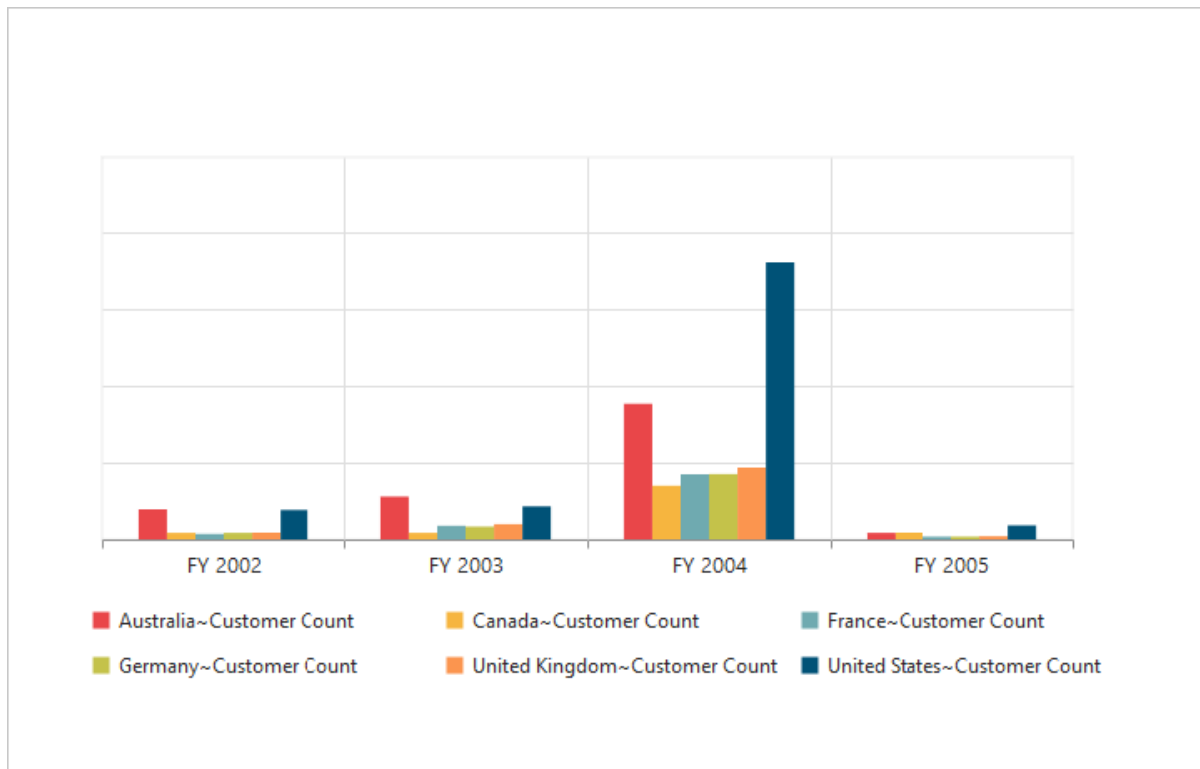
#### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        primaryYAxis: {
            visible: false
        }
    });
});

```

```
},
});
});
```



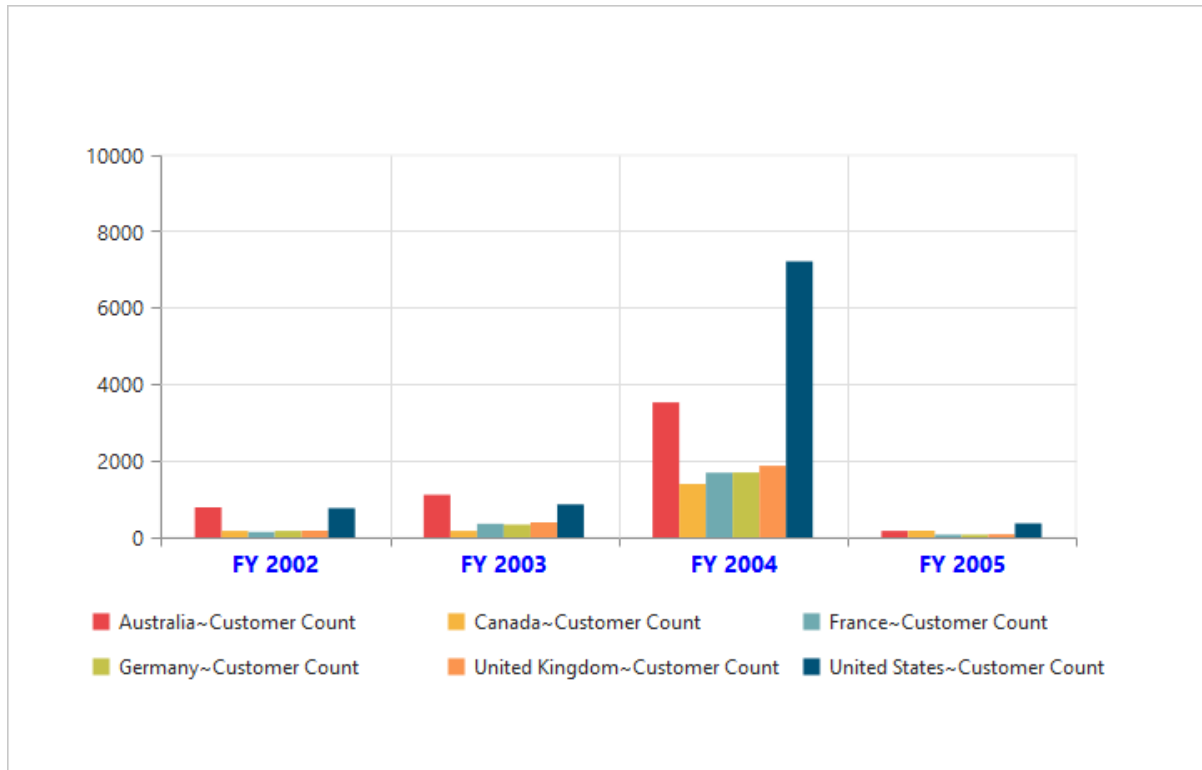
#### Label customization

By using the `font` property of the axis, you can customize the font family, color, opacity, size, and font-weight of labels.

#### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
primaryXAxis: {
font: {
size: "20px",
color: "blue",
fontWeight: "bold",
fontFamily: "Segoe UI"
}
},
},
});
});
```





### Label and tick positioning

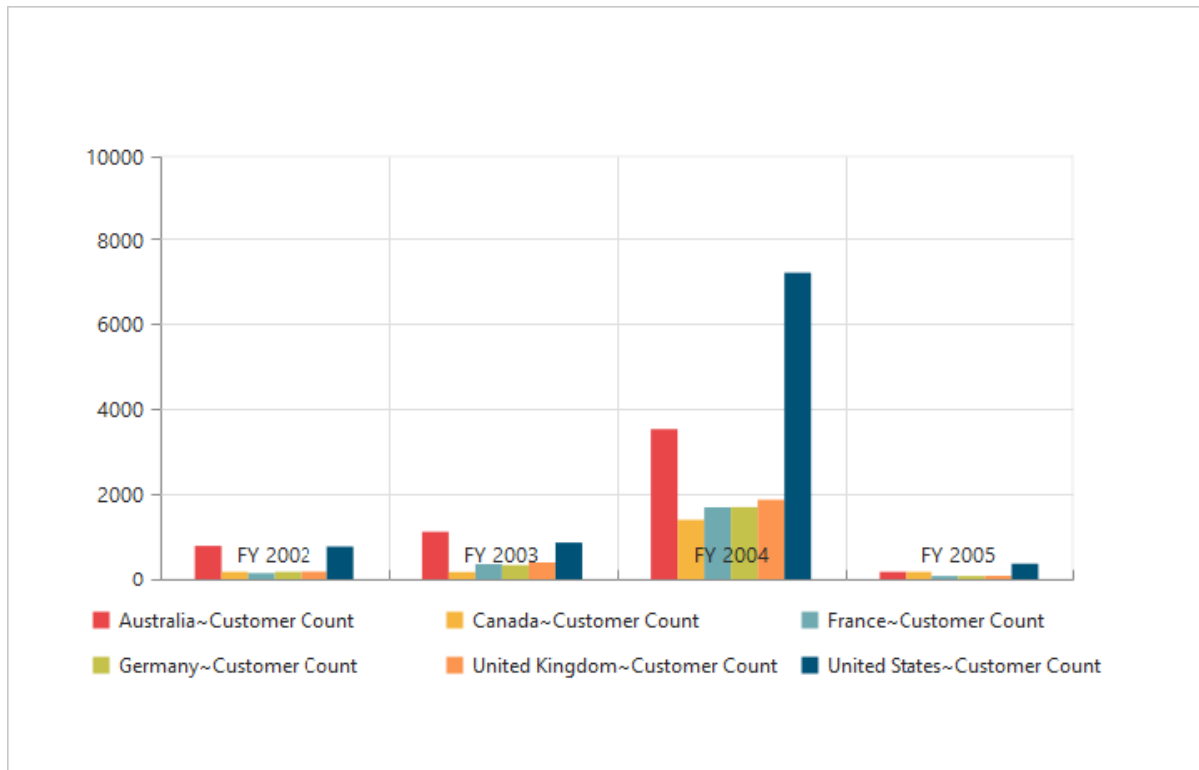
Axis labels and ticks can be positioned inside or outside the chart area by using the `axisLabelPosition` and `tickLinesPosition` properties. The labels and ticks are positioned outside the chart area, by default.

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
primaryXAxis: {
axisLabelPosition: "inside",
tickLinesPosition: "inside",
},
});
});

```



### Grid lines customization

By using the `majorGridLines` and `minorGridLines` properties of the axis, you can customize the width, color, visibility, and opacity of the grid lines.

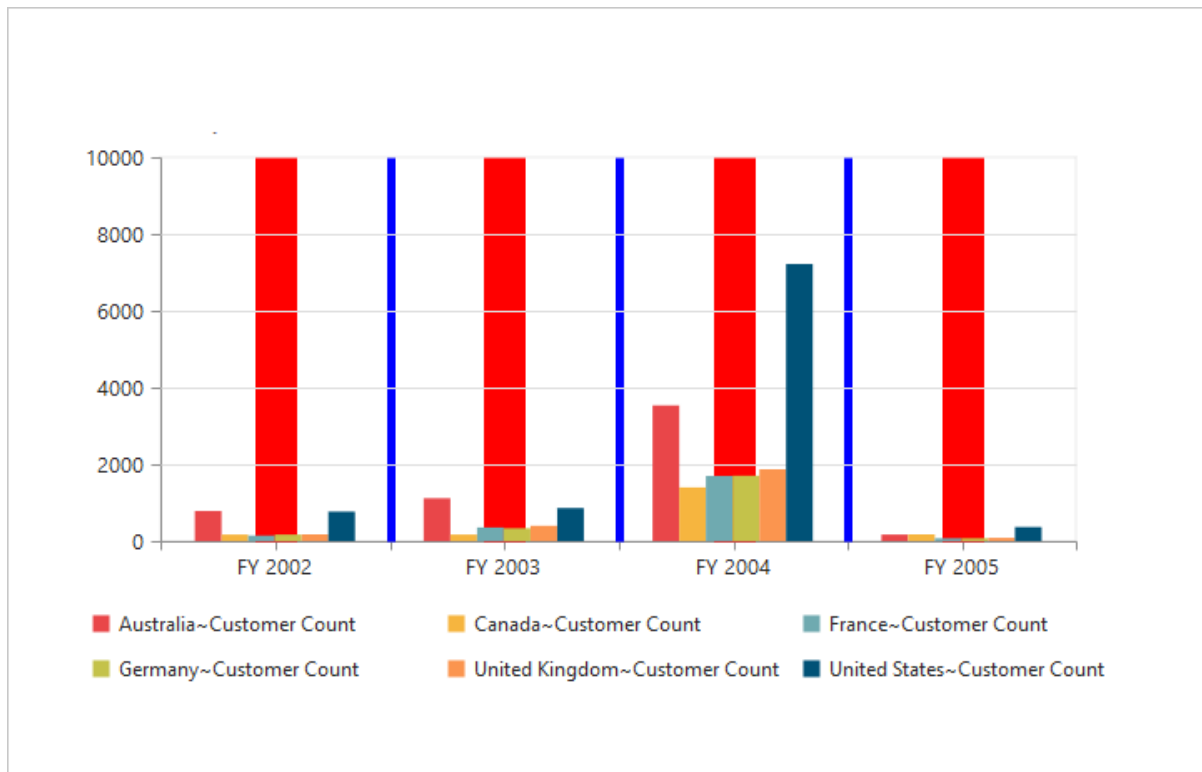
**Note:** By default, the minor grid lines are not visible in the pivot chart.

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
primaryXAxis: {
majorGridLines: {
width: 5,
color: 'blue',
visible:true
},
minorGridLines: {
width: 25,
color: 'red',
visible: true
},
minorTicksPerInterval: 1
},
});
});

```



#### Tick line customization

By using the `majorTickLines` and `minorTickLines` properties of the axis, you can customize the width, color, visibility, size, and opacity of the tick lines.

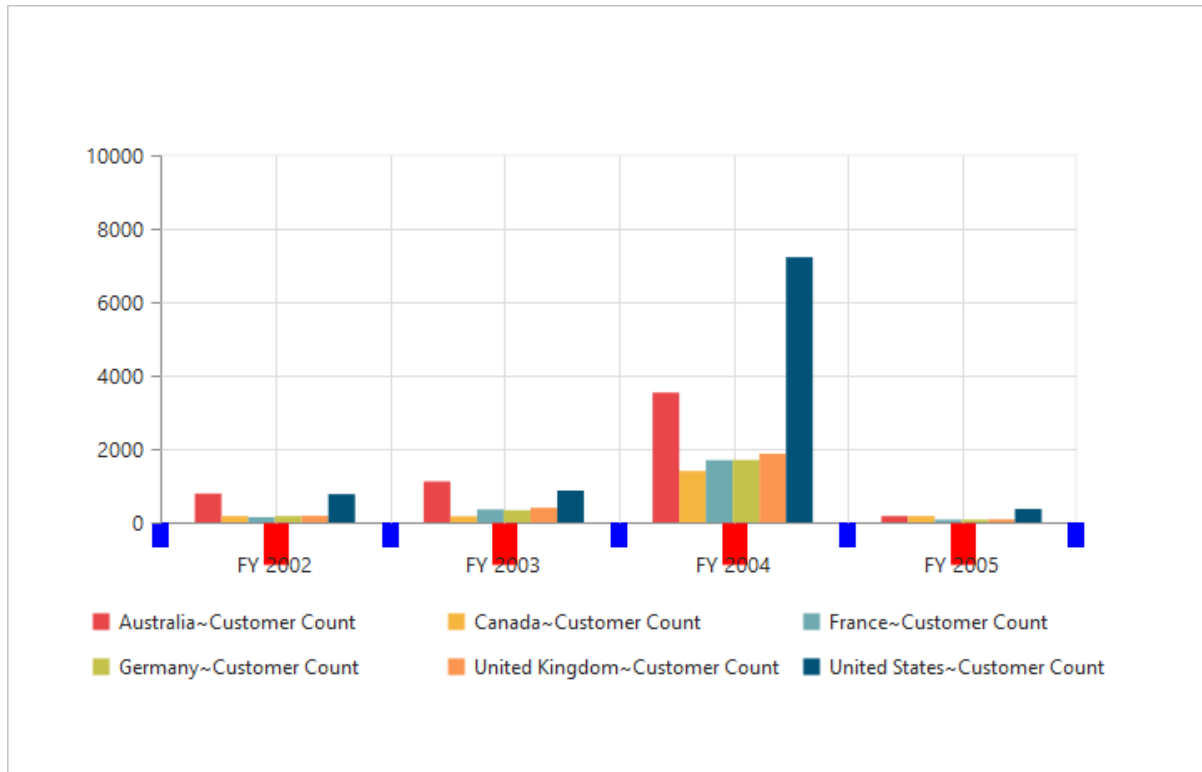
**Note:** By default, the minor tick lines are not visible in the pivot chart.

#### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
primaryXAxis: {
majorTickLines: {
width: 10,
size: 15,
color: 'blue',
visible:true
},
minorTickLines: {
width: 15,
size: 25,
color: 'red',
visible: true
},
minorTicksPerInterval: 1
},
});
});

```



### Inverting axis

The axis can be inverted by using the `isInversed` property of the axis.

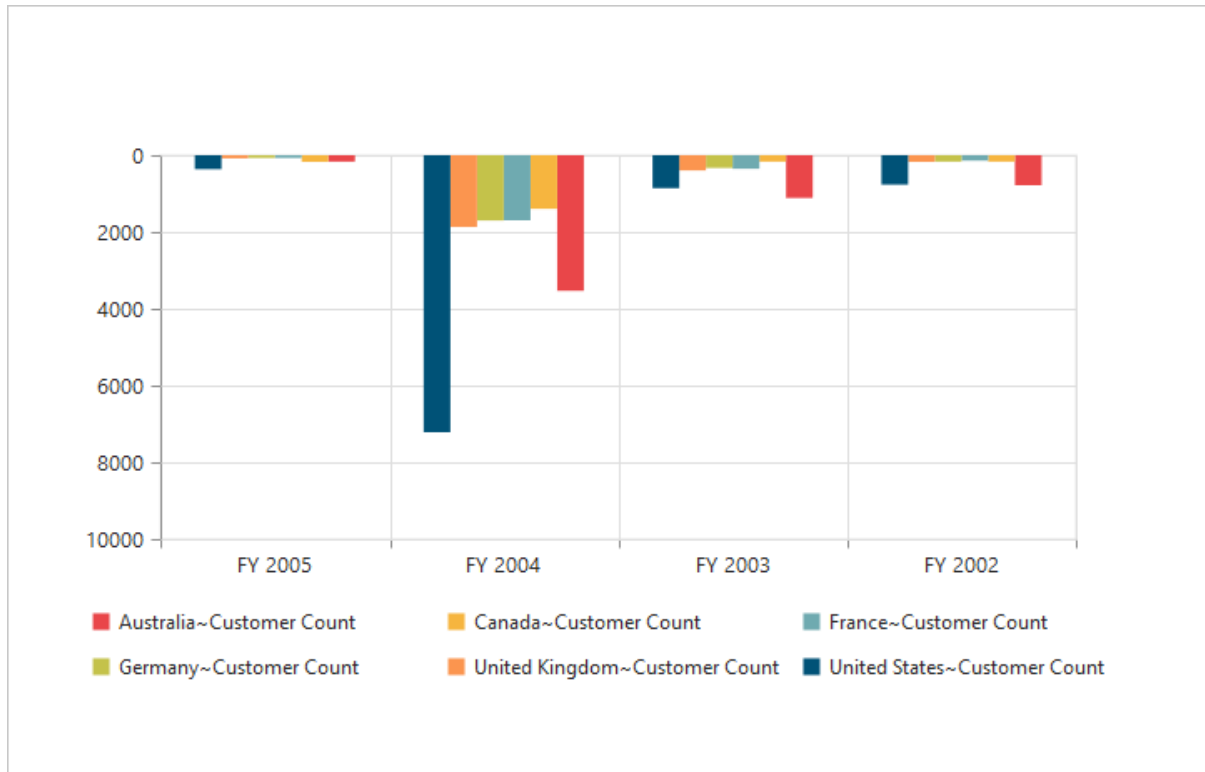
**Note:** By default, the `isInversed` property is false in the pivot chart.

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
primaryYAxis: {
isInversed: true
},
primaryXAxis: {
isInversed: true
},
});
});

```



### Placing axes at opposite side

The `opposedPosition` property of chart axis can be used to place the axis at the opposite direction from its default position.

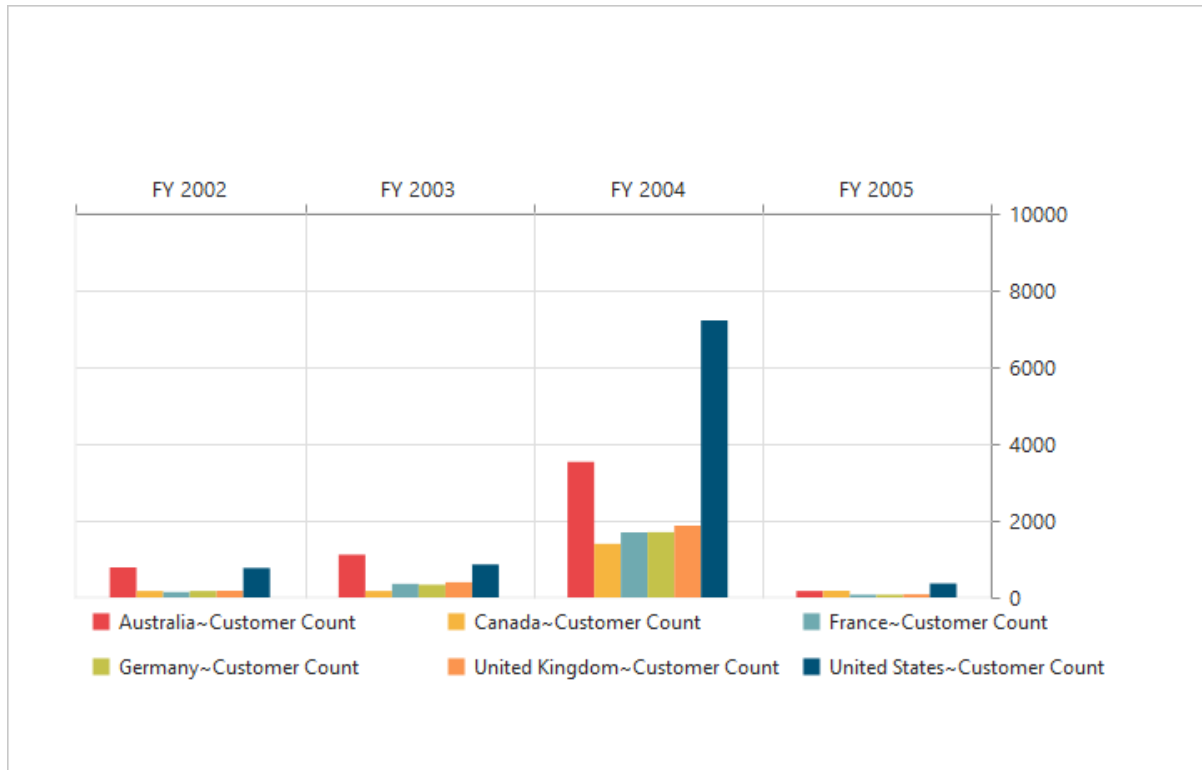
**Note:** By default, the `opposedPosition` property is false in the pivot chart.

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
primaryYAxis: {
opposedPosition: true
},
primaryXAxis: {
opposedPosition: true
},
});
});

```



### Multi-level labels

Multi-level labels allows you to drill down to access the detailed level of data or drill up to see the summarized data by using the expander present in the OLAP chart.

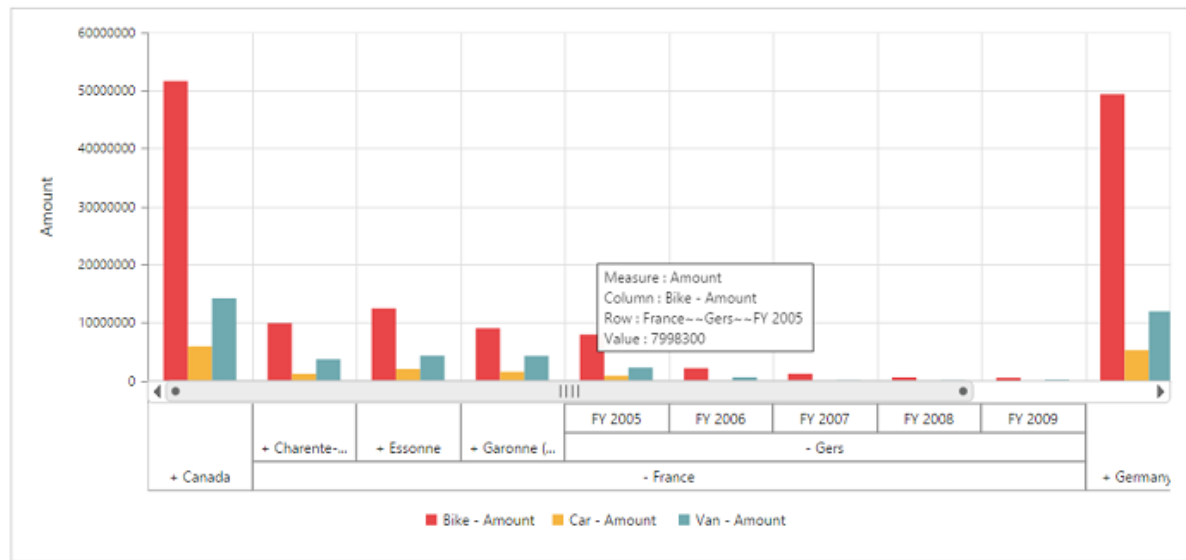
### HTML

```

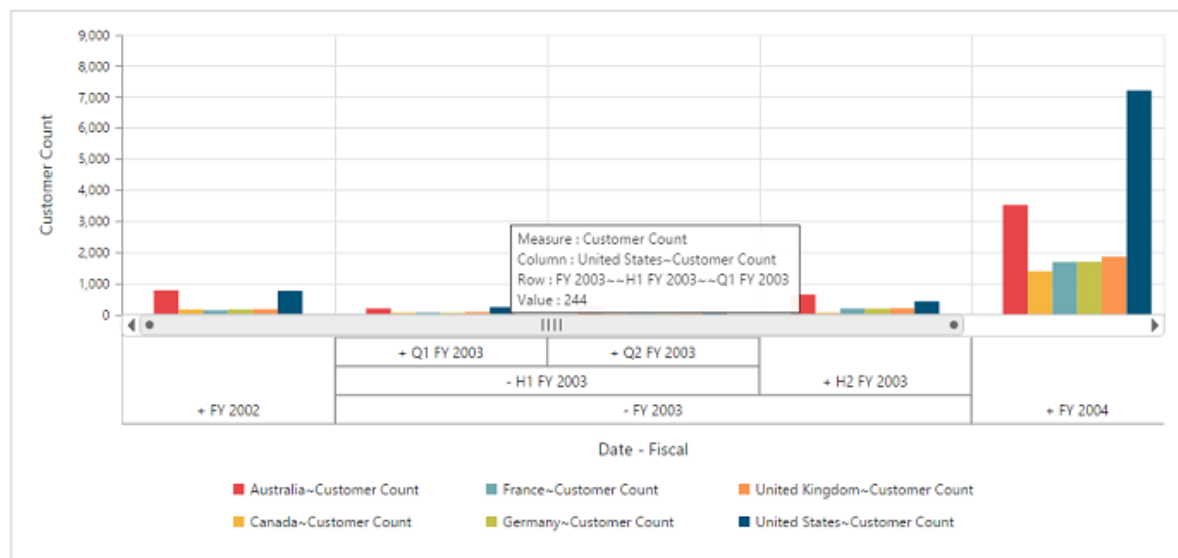
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
enableMultiLevelLabels: true
});
});

```

## Relational



## OLAP



## Multiple Axes

You can split the pivot chart and its area into multiple panes to draw multiple series with multiple axes. Additional vertical/horizontal axes in the pivot chart can be created by the [axes](#) property.

Before rendering multiple axes in the pivot chart, you should know some important properties of axes and series of the pivot chart.

*Properties of axes*

- **rowIndex:** Specifies the index of the row where the axis is associated.
- **columnIndex:** Specifies the index of the column where the axis is associated.
- **opposedPosition:** Specifies whether to render the axis at the opposite side of its default position.

- **labelFormat**: Provides custom formatting for axis label and supports all standard formatting types of numerical and date time values.
- **title**: You can customize the title of the axis by the following properties.
- **enableTrim**: Specifies whether to trim the axis title when it exceeds the chart area or the maximum width of the title.
- **maximumTitleWidth**: When the title exceeds the maximum width, the title gets trimmed by setting enableTrim to true.
- **visible**: Controls the visibility of the axis title.
- **name**: Unique name of the axis. To associate an axis with the series, you have to set this name to the xAxisName/yAxisName property of the series.

To know more about axes properties, [Click Here](#)

#### *Properties of series*

- **xAxisName**: Specifies the name of the X-axis that has to be associated with this series. Add an axis instance with this name to axes collection.
- **yAxisName**: Specifies the name of the Y-axis that has to be associated with this series. Add an axis instance with this name to axes collection.

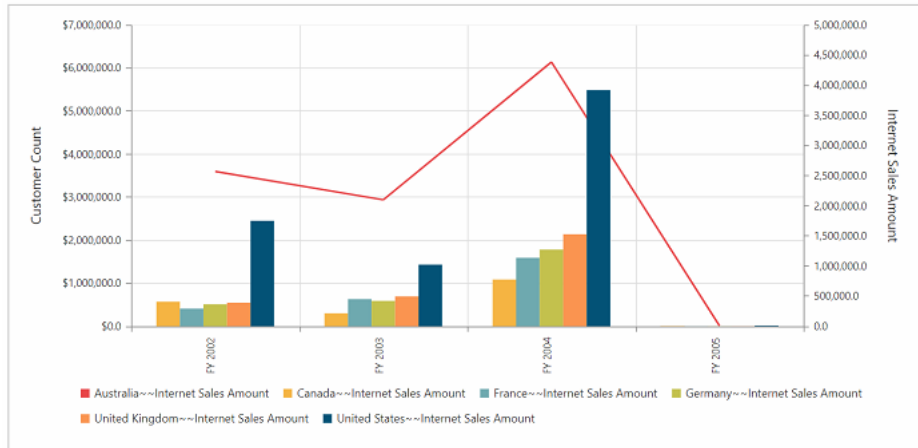
#### Customizing axes at row index of zero

You can customize the axes in the primary pivot chart itself. To achieve this, you should give the following properties:

#### **JAVASCRIPT**

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
axes:[{
rowIndex: 0,
name: 'yAxisConfig',
//...
}],
beforeSeriesRender : "onBeforeRender"
});
function onBeforeRender(args) {
for (var i = 0; i < args.series.length; i++) {
if (args.series[i].name.indexOf("Australia") != -1) {
args.series[i].yAxisName = "yAxisConfig";
args.series[i].type = "line";
}
}
return args;
}
});
```





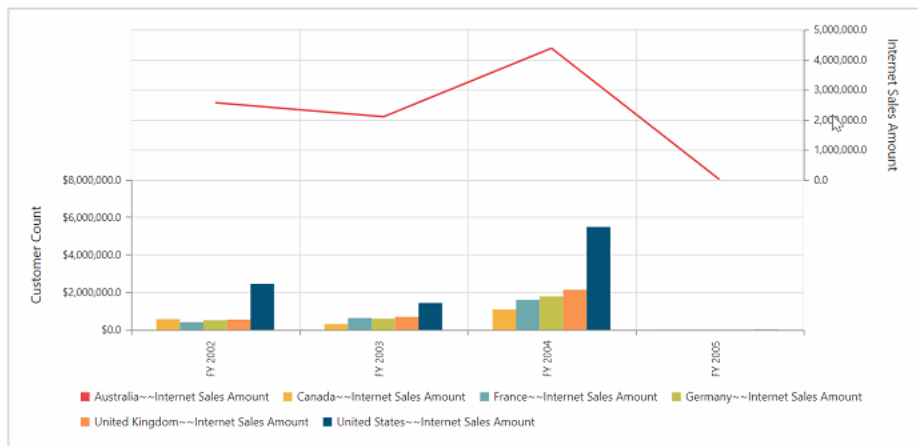
Customizing axes at row index one

### JAVASCRIPT

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        axes: [{
            rowIndex: 1,
            name: 'yAxisConfig',
            //...
        }],
        beforeSeriesRender : "onBeforeRender"
    });
    function onBeforeRender(args) {
        for (var i = 0; i < args.series.length; i++) {
            if (args.series[i].name.indexOf("Australia") !== -1) {
                args.series[i].yAxisName = "yAxisConfig";
                args.series[i].type = "line";
            }
        }
        return args;
    }
});

```



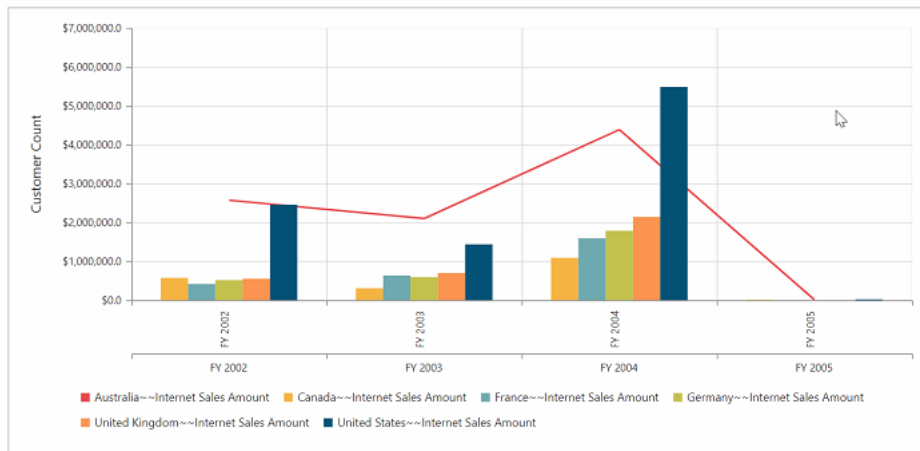
Customizing axes at column index of zero

### JAVASCRIPT

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
axes:[{
columnIndex: 0,
name: 'xAxisConfig',
//...
}],
beforeSeriesRender : "onBeforeRender"
});
function onBeforeRender(args) {
for (var i = 0; i < args.series.length; i++) {
if (args.series[i].name.indexOf("Australia") != -1) {
args.series[i].xAxisName = "xAxisConfig";
args.series[i].type = "line";
}
}
return args;
}
});

```



Customizing axes at column index of one

### JAVASCRIPT

```

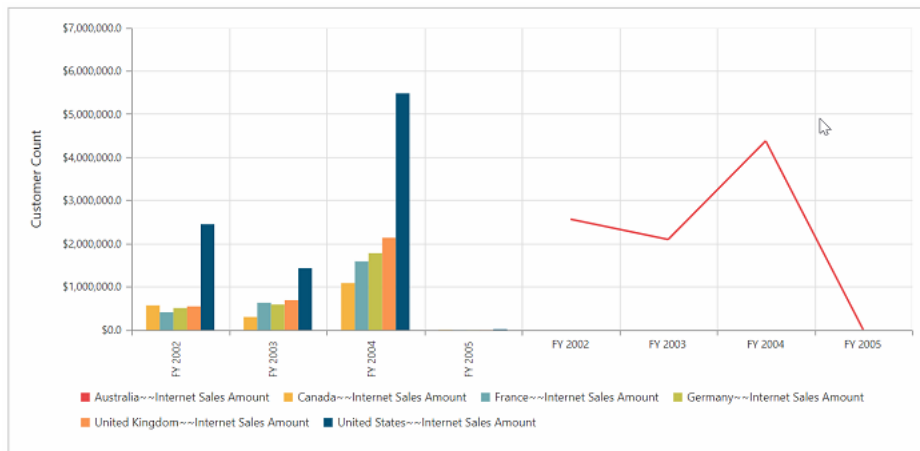
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
axes:[{
columnIndex: 1,
name: 'xAxisConfig',
//...
}],
beforeSeriesRender : "onBeforeRender"
});
function onBeforeRender(args) {

```

```

for (var i = 0; i < args.series.length; i++) {
    if (args.series[i].name.indexOf("Australia") != -1) {
        args.series[i].xAxisName = "xAxisConfig";
        args.series[i].type = "line";
    }
}
return args;
}
});

```



### Customizing series

You can customize the series in multiple axes support with the help of **beforeSeriesRender** event. You can change the series type through the **onBeforeRender** event.

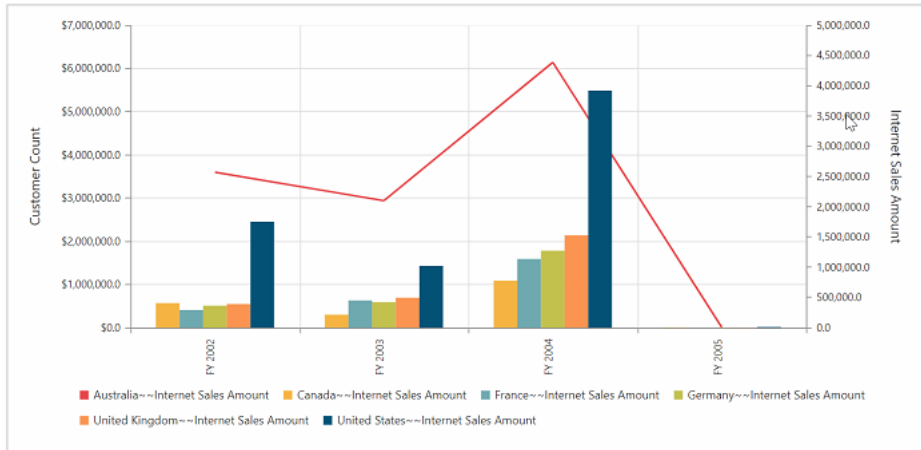
### JAVASCRIPT

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        axes:[{
            name: 'yAxisConfig',
            //...
        }],
        beforeSeriesRender : "onBeforeRender"
    });
    function onBeforeRender(args) {
        for (var i = 0; i < args.series.length; i++) {
            if (args.series[i].name.indexOf("Australia") != -1) {
                args.series[i].yAxisName = "yAxisConfig";
                args.series[i].type = "line";
            }
        }
        return args;
    }
});

```

**Note:** You have to use the same name in both name property of axes and xAxisName/yAxisName property of series in the above **beforeSeriesRender** event.



To learn more about series properties, [click here](#).

### Multiple axes support by series index

You can render the pivot chart with multiple axes by series index.

**Note:** This is the default behavior of multiple axes support, if you are not triggering the `beforeSeriesRender` event.

### JAVASCRIPT

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
axes: [
{
name: 'y:0' // you can create separate y axes for the series which you want
// y indicates(yAxisName) axis in which series needs to be added.
// 0 indicates the series index of pivot chart.
// you can also use pass multiple series index separated by comma(y:0,2)
},
{
name: 'x:1' //you can create separate x axes for the series which you want
// x indicates(xAxisName) axis in which series needs to be added.
// 0 indicates the series index of pivot chart.
// you can also use pass multiple series index separated by comma(x:0,2)
}
]
});
});

```

For Y-axes

### JAVASCRIPT

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
axes: [
{
name: 'y:0'

```

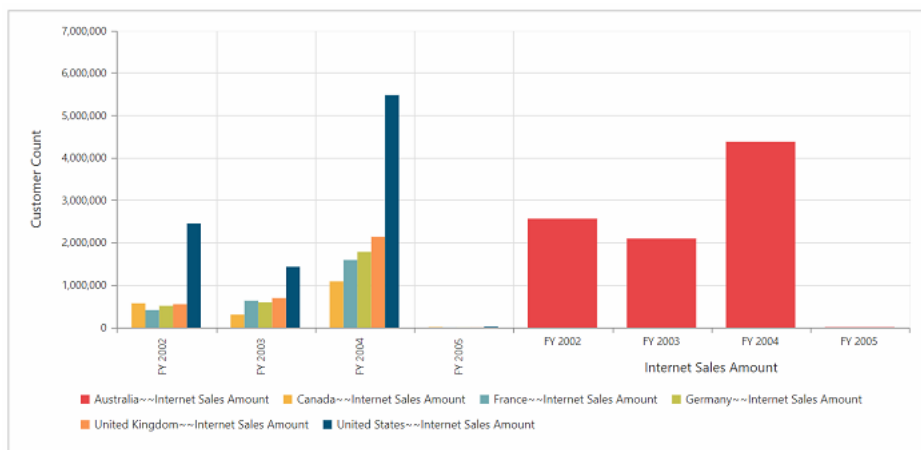
```
}]
});
});
```



For X-axes

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotChart($("#PivotChart1"), {
        axes: [
            {
                name: 'x:0'
            }
        ]
    });
});
```



Customizing PrimaryYAxis and axes properties

*labelFormat*

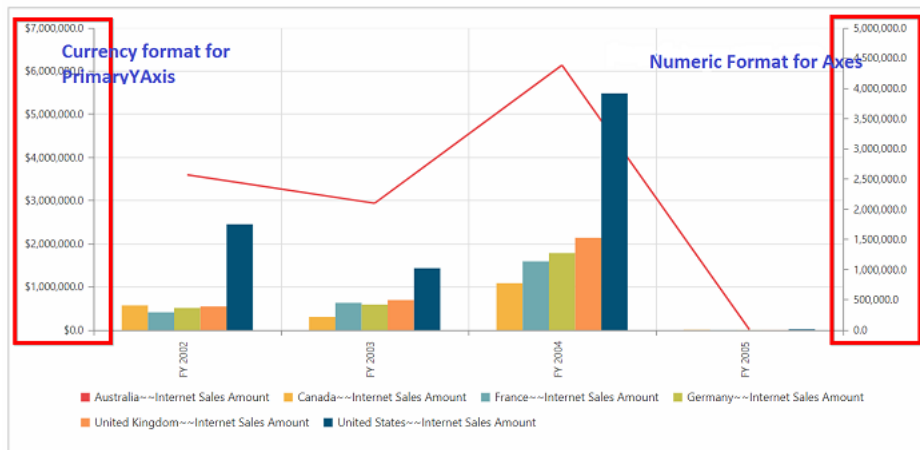
You can customize the **labelFormat** for both PrimaryYAxis and custom axes.

### JAVASCRIPT

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
axes:[{
//...
labelFormat:'n1'
}],
primaryYAxis: { labelFormat: 'c1' }
});
});

```



### title

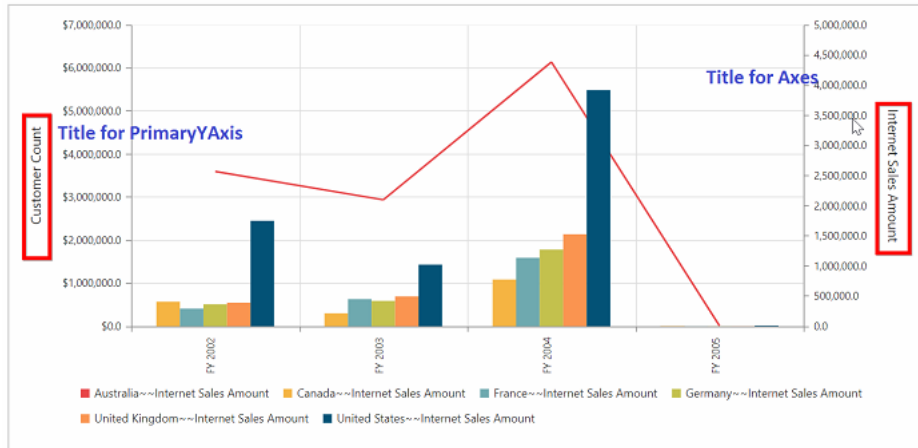
You can customize the title for axes by the **title** property.

### JAVASCRIPT

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
axes:[{
//...
title: {text: "Internet Sales Amount"},
}],
primaryYAxis: { title: { text: "Customer Count" }}
});
});

```



## Exporting

The pivot chart control can be exported to the following file formats:

- Microsoft Excel
- Microsoft Word
- PDF
- Image

The pivot chart control can be exported by invoking **"exportPivotChart"** method with an appropriate export option as a parameter.

### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="PivotChart1" style="min-height: 275px; min-width: 525px; height: 460px; width: 100%"></div>
<button id="button"></button>
<div>
</body>
</html>
```

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.Column
}
});
var basicButton = new ej.Button($("#button"), {
size: "large",
showRoundedCorner: true,
text: "Export",
click: function (args) {
let chartObj = $('<div>e-pivotchart</div>').data("ejPivotChart");
```

```
chartObj.exportPivotChart("http://js.syncfusion.com/ejservices/api/PivotChart/Olap/ExcelExport", "fileName");
}
});
});
```

### Excel export

You can export the contents of the pivot chart to Excel document for future archival, references, and analysis purposes.

To achieve Excel export, the service URL and the file name are set as parameters.

#### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="PivotChart1" style="min-height: 275px; min-width: 525px; height: 460px; width: 100%"></div>
<button id="button"></button>
<div>
</body>
</html>
```

#### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.Column
}
});
var basicButton = new ej.Button($("#button"), {
size: "large",
showRoundedCorner: true,
text: "Export",
click: function (args) {
let chartObj = $('<e-pivotchart>').data("ejPivotChart");
chartObj.exportPivotChart("http://js.syncfusion.com/ejservices/api/PivotChart/Olap/ExcelExport", "fileName");
}
});
});
```

### Word export

You can export the contents of the pivot chart to a Word document for future archival, references, and analysis purposes.

To achieve Word export, the service URL and the file name are set as parameters.

#### HTML

```
<!DOCTYPE html>
```



```
<html>
<body>
<div id="PivotChart1" style="min-height: 275px; min-width: 525px; height: 460px; width: 100%"></div>
<button id="button"></button>
<div>
</body>
</html>
```

**HTML**

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.Column
}
});
var basicButton = new ej.Button($("#button"), {
size: "large",
showRoundedCorner: true,
text: "Export",
click: function (args) {
let chartObj = $('<div>e-pivotchart</div>').data("ejPivotChart");
chartObj.exportPivotChart("http://js.syncfusion.com/ejservices/api/PivotChart/Olap/WordExport", "fileName");
}
});
});
```

**PDF export**

You can export the contents of the pivot chart to a PDF document for future archival, references, and analysis purposes.

To achieve PDF export, the service URL and the file name are set as parameters.

**HTML**

```
<!DOCTYPE html>
<html>
<body>
<div id="PivotChart1" style="min-height: 275px; min-width: 525px; height: 460px; width: 100%"></div>
<button id="button"></button>
<div>
</body>
</html>
```

**HTML**

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
```

```

commonSeriesOptions: {
  type: ej.PivotChart.ChartTypes.Column
}
});
var basicButton = new ej.Button($("#button"), {
  size: "large",
  showRoundedCorner: true,
  text: "Export",
  click: function (args) {
    let chartObj = $('.e-pivotchart').data("ejPivotChart");
    chartObj.exportPivotChart("http://js.syncfusion.com/ejservices/api/PivotChart/Olap/PDFExport", "fileName");
  }
});
});
});

```

### Image export

You can export contents of the pivot chart to image format for future archival, references, and analysis purposes. You can export the pivot chart to the following image formats:

- PNG
- EMF
- JPG
- GIF
- BMP

To export pivot chart in PNG format, the service URL, file name, and **“ej.PivotChart.ExportOptions.PNG”** enumeration value are sent as parameters. This is similar to other image formats.

### HTML

```

<!DOCTYPE html>
<html>
<body>
<div id="PivotChart1" style="min-height: 275px; min-width: 525px; height: 460px; width: 100%"></div>
<button id="button"></button>
<div>
</body>
</html>

```

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
  var sample = new ej.PivotChart($("#PivotChart1"), {
    commonSeriesOptions: {
      type: ej.PivotChart.ChartTypes.Column
    },
  });
  var basicButton = new ej.Button($("#button"), {
    size: "large",
    showRoundedCorner: true,

```

```

text: "Export",
click: function (args) {
let chartObj = $('e-pivotchart').data("ejPivotChart");
chartObj.exportPivotChart("http://js.syncfusion.com/ejservices/api/PivotChart/Olap/ImageExport", "fileName", "png");
}
});
});

```

### Exporting customization

You can add the title and description to the exporting document by using the title and description property obtained in the "beforeExport" event.

---

**Note:** Title and description cannot be added to image formats.

---

### HTML

```

<!DOCTYPE html>
<html>
<body>
<div id="PivotChart1" style="min-height: 275px; min-width: 525px; height: 460px; width: 100%"></div>
<button id="button"></button>
<div>
</body>
</html>

```

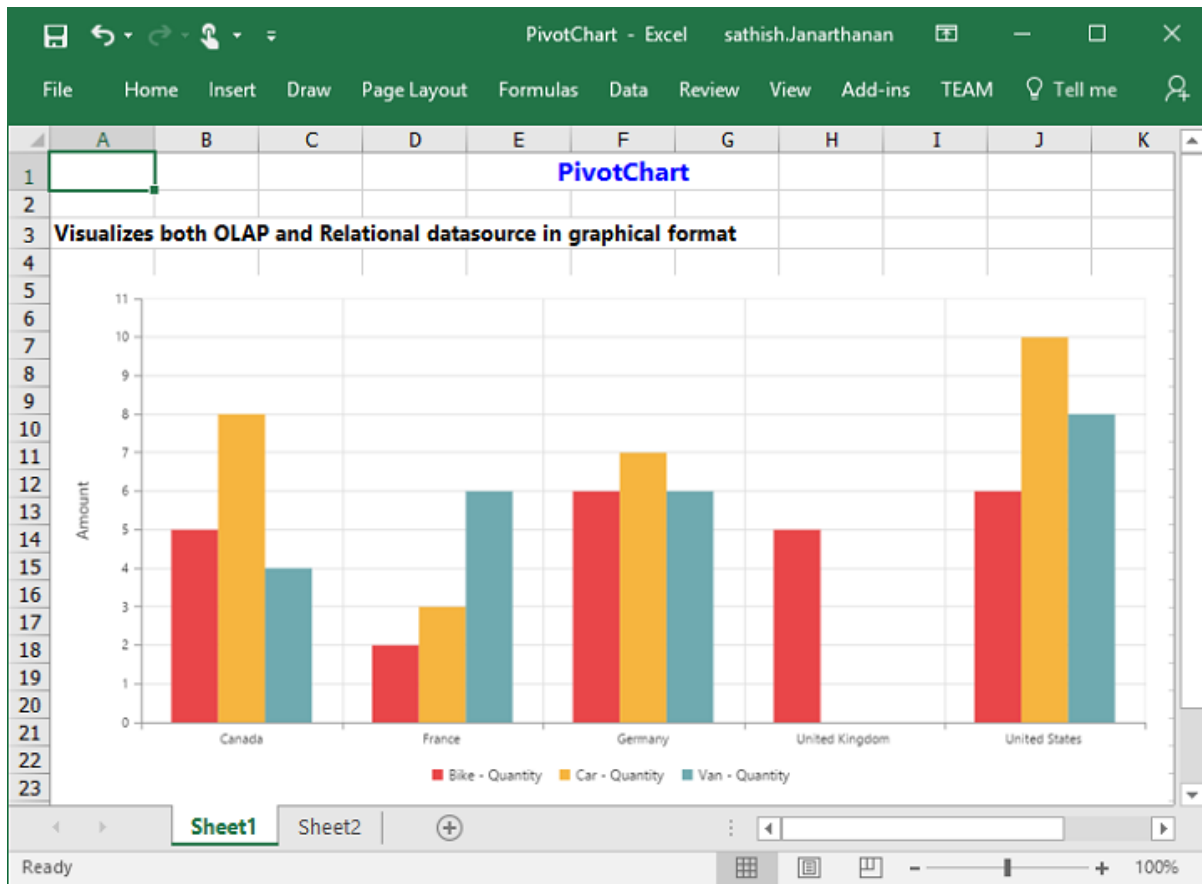
### HTML

```

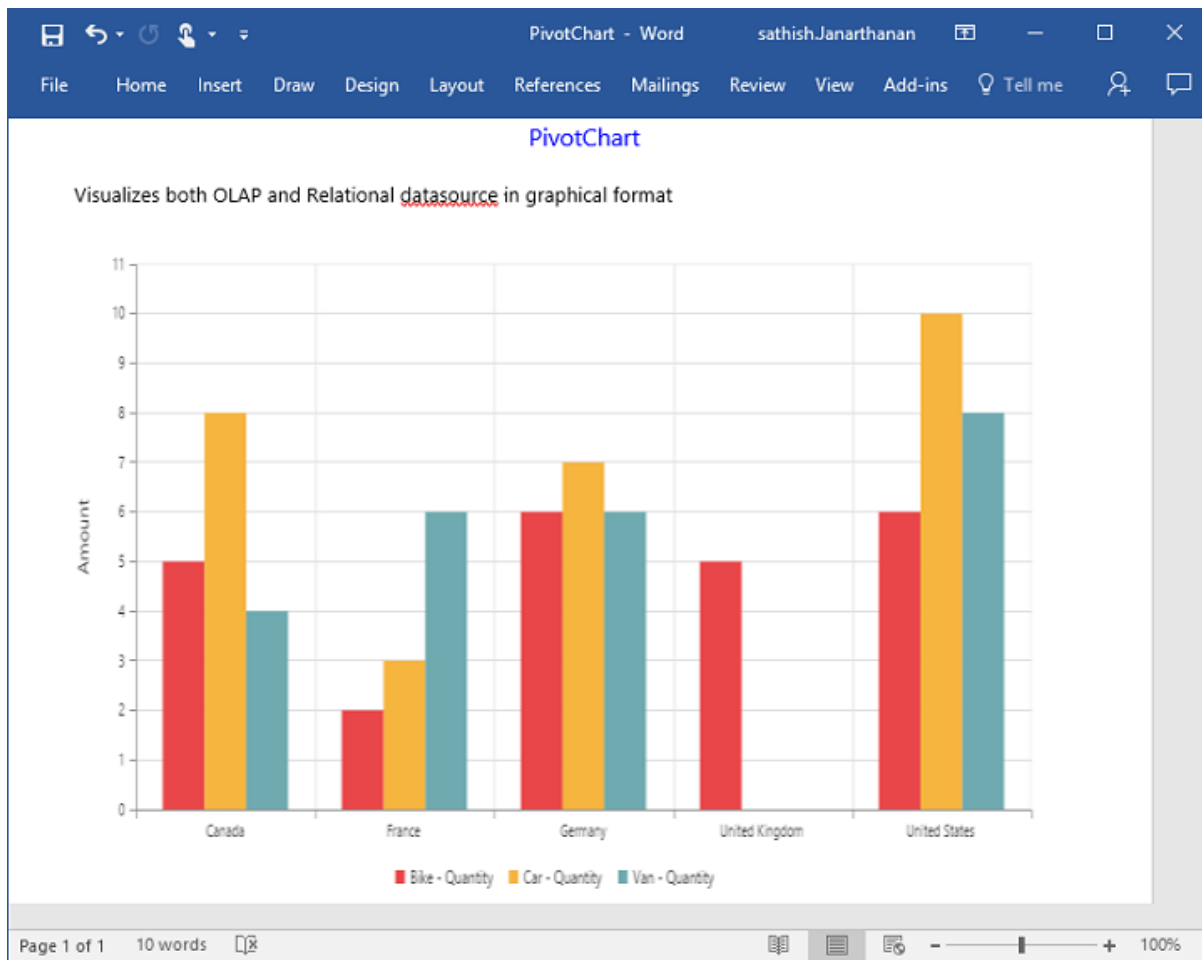
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotChart($("#PivotChart1"), {
commonSeriesOptions: {
type: ej.PivotChart.ChartTypes.Column
},
beforeExport: function(args) {
args.title = "PivotChart";
args.description = "Visualizes both OLAP and Relational datasource in graphical format";
}
});
var basicButton = new ej.Button($("#button"), {
size: "large",
showRoundedCorner: true,
text: "Export",
click: function (args) {
let chartObj = $('e-pivotchart').data("ejPivotChart");
chartObj.exportPivotChart("http://js.syncfusion.com/ejservices/api/PivotChart/Olap/ExcelExport", "fileName");
}
});
});

```

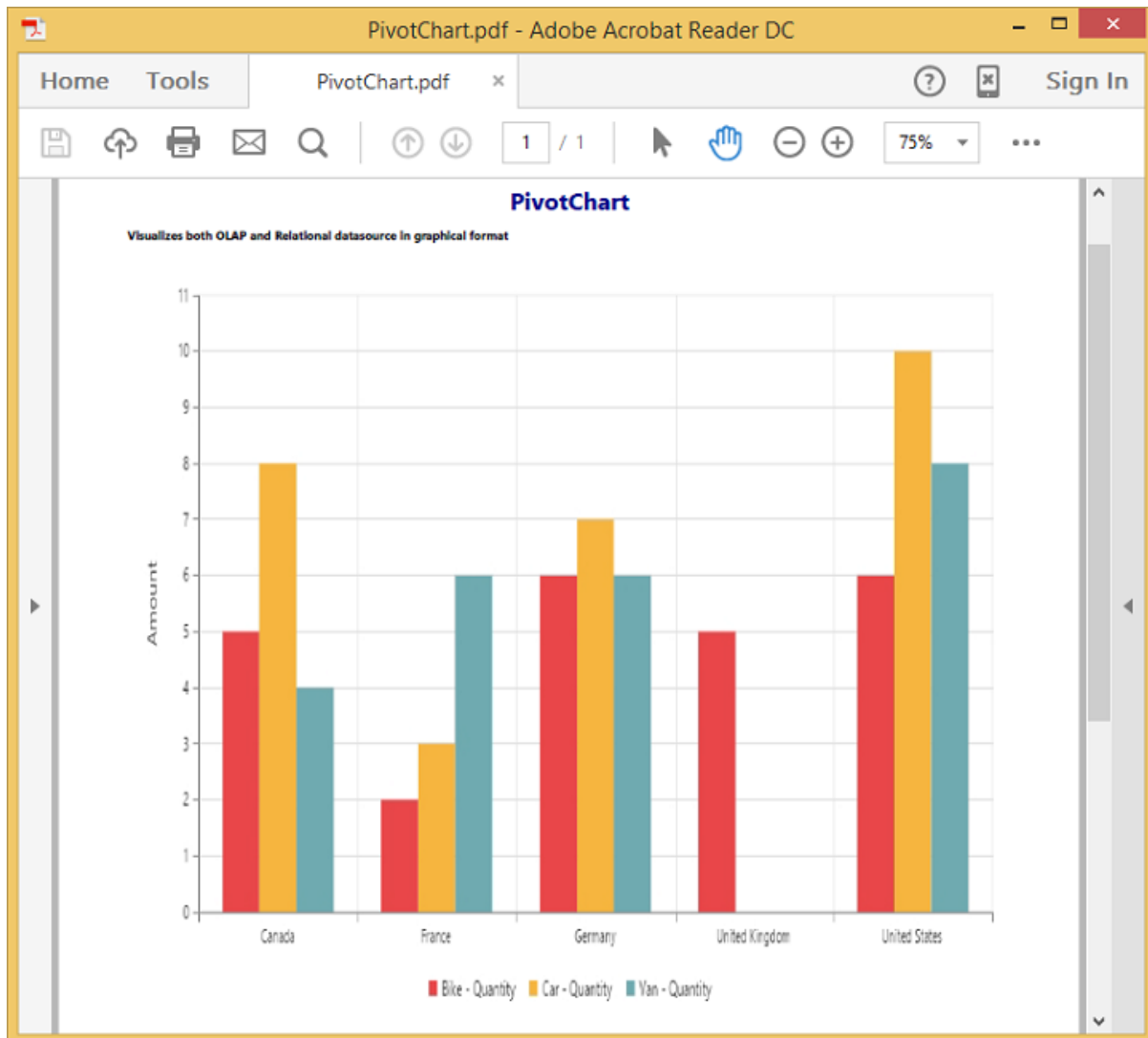
The following screenshot shows the pivot chart control exported to an Excel document:



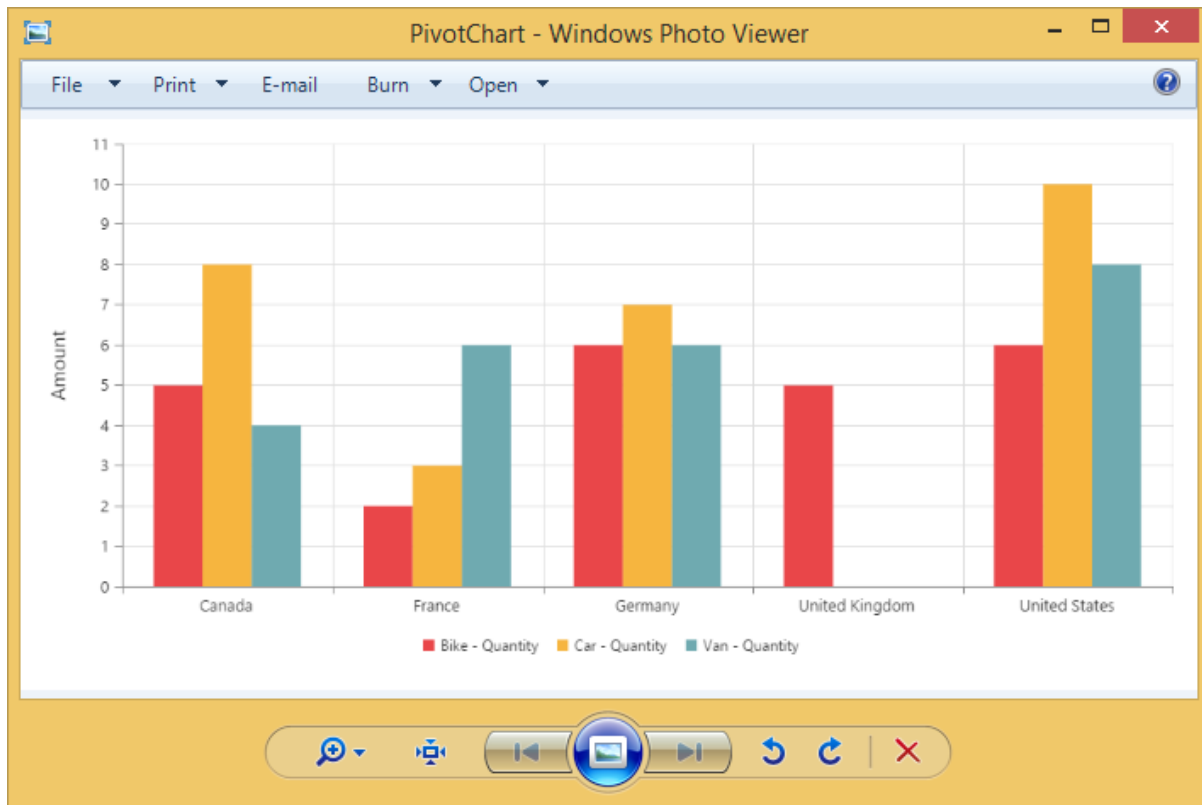
The following screenshot shows the pivot chart control exported to a Word document:



The following screenshot shows the pivot chart control exported to a PDF document:



The following screenshot shows the pivot chart control exported to a PNG format:



## PivotGrid

### Overview

The **PivotGrid** control is an easily configurable, presentation-quality business control that reads both OLAP and Relational datasource, allows to create multi-dimensional views for analysis and satisfying business user needs.

### Key Features

Some of the key features of PivotGrid are as follows,

- **Data Binding:** Supports both OLAP and Relational datasource data binding in the same environment.
- **PivotTable Field List:** Lists the entire datasource bound to PivotGrid and allows UI interaction such as filtering and drag drop operation of any measure, dimension, hierarchy, level and field at runtime.
- **Grouping Bar:** Supports UI interaction at runtime through sorting, filtering and drag drop operation of the existing field items bound through report.
- **Grid Layout:** Four different layouts such as normal, top summary, no summaries and excel-like layout are supported.
- **Export:** PivotGrid can be exported to Word, PDF, CSV and Excel documents.
- PivotGrid control specific features such as hyperlink, cell selection, conditional formatting, RTL, globalization and localization are supported as well.

## Getting Started

This section explains you the steps required to populate the PivotGrid with data source. This section covers only the minimal features that you need to know to get started with the PivotGrid.

For common steps of typescript , you can refer [here](#).

The default type definition file ej.web.all.d.ts needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the intelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the jquery.d.ts file in your TypeScript application, which can be downloaded from [here](#).

## Script and CSS Reference

Add the scripts and CSS references to the “index.html” page as the order mentioned in the following code example.

### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
<script
src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.valida
te.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

In the above code, `ej.web.all.min.js` script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

## Relational

This section covers the information that you need to know to populate a simple PivotGrid with Relational data source.



### Control Initialization

Add a `div` container to render the PivotGrid.

#### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="PivotGrid1" style="overflow: auto"></div>
</body>
</html>
```

Initialize the PivotGrid in `app.ts` file by using the `ej.PivotGrid` method.

#### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotGrid($("#PivotGrid1"), { });
});
```

### Populate PivotGrid with Data

Let us now see how to populate the PivotGrid control using a sample JSON data as shown below.

#### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
var pivot_dataset = [
{ Amount: 100, Country: "Canada", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Alberta" },
{ Amount: 200, Country: "Canada", Date: "FY 2006", Product: "Van",
Quantity: 3, State: "British Columbia" },
{ Amount: 300, Country: "Canada", Date: "FY 2007", Product: "Car",
Quantity: 4, State: "Brunswick" },
{ Amount: 150, Country: "Canada", Date: "FY 2008", Product: "Bike",
Quantity: 3, State: "Manitoba" },
{ Amount: 200, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 4, State: "Ontario" },
{ Amount: 100, Country: "Canada", Date: "FY 2007", Product: "Van",
Quantity: 1, State: "Quebec" },
{ Amount: 200, Country: "France", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Charente-Maritime" },
{ Amount: 250, Country: "France", Date: "FY 2006", Product: "Van",
Quantity: 4, State: "Essonne" },
{ Amount: 300, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 3, State: "Garonne (Haute)" },
{ Amount: 150, Country: "France", Date: "FY 2008", Product: "Van",
Quantity: 2, State: "Gers" },
{ Amount: 200, Country: "Germany", Date: "FY 2006", Product: "Van",
Quantity: 3, State: "Bayern" },
{ Amount: 250, Country: "Germany", Date: "FY 2007", Product: "Car",
Quantity: 3, State: "Brandenburg" },
{ Amount: 150, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 4, State: "Hamburg" },
```

```
{ Amount: 200, Country: "Germany", Date: "FY 2008", Product: "Bike",
Quantity: 4, State: "Hessen" },
{ Amount: 150, Country: "Germany", Date: "FY 2007", Product: "Van",
Quantity: 3, State: "Nordrhein-Westfalen" },
{ Amount: 100, Country: "Germany", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Saarland" },
{ Amount: 150, Country: "United Kingdom", Date: "FY 2008", Product:
"Bike", Quantity: 5, State: "England" },
{ Amount: 250, Country: "United States", Date: "FY 2007", Product: "Car",
Quantity: 4, State: "Alabama" },
{ Amount: 200, Country: "United States", Date: "FY 2005", Product: "Van",
Quantity: 4, State: "California" },
{ Amount: 100, Country: "United States", Date: "FY 2006", Product:
"Bike", Quantity: 2, State: "Colorado" },
{ Amount: 150, Country: "United States", Date: "FY 2008", Product: "Car",
Quantity: 3, State: "New Mexico" },
{ Amount: 200, Country: "United States", Date: "FY 2005", Product:
"Bike", Quantity: 4, State: "New York" },
{ Amount: 250, Country: "United States", Date: "FY 2008", Product: "Car",
Quantity: 3, State: "North Carolina" },
{ Amount: 300, Country: "United States", Date: "FY 2007", Product: "Van",
Quantity: 4, State: "South Carolina" }
];
$(function () {
var sample = new ej.PivotGrid($("#PivotGrid1"), {
dataSource: {
data: pivot_dataset,
rows: [
{
fieldName: "Country",
fieldCaption: "Country"
}
],
columns:
[
{
fieldName: "Product",
fieldCaption: "Product"
}
],
values: [
{
fieldName: "Amount",
fieldCaption: "Amount"
}
],
filters: []
}
});
});
```

The above code will generate a simple PivotGrid with “Country” field in Row, “Product” field in Column and “Amount” field in Value section.

|                | Bike   | Car    | Van    | Grand Total |
|----------------|--------|--------|--------|-------------|
|                | Amount | Amount | Amount | Amount      |
| Canada         | 250    | 500    | 300    | 1050        |
| France         | 200    | 300    | 400    | 900         |
| Germany        | 300    | 400    | 350    | 1050        |
| United Kingdom | 150    |        |        | 150         |
| United States  | 300    | 650    | 500    | 1450        |
| Grand Total    | 1200   | 1850   | 1550   | 4600        |

## OLAP

This section covers the information that you need to know to populate a simple PivotGrid with OLAP data source.

### Control Initialization

Add a `div` container to render the PivotGrid.

### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="PivotGrid1" style="overflow: auto"></div>
</body>
</html>
```

Initialize the PivotGrid in ts file by using the `ej.PivotGrid` method.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotGrid($("#PivotGrid1"), { });
});
```

### Populate PivotGrid with data

Let us now see how to populate the PivotGrid control using a sample JSON data as shown below.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotGrid($("#PivotGrid1"), {
dataSource: {
data: "http://bi.syncfusion.com/olap/msmdpump.dll",
catalog: "Adventure Works DW 2008 SE",
cube: "Adventure Works",
rows: [
```

```
{
  fieldName: "[Date].[Fiscal]"
},
columns: [
  {
    fieldName: "[Customer].[Customer Geography]"
  }
],
values: [
  {
    measures: [
      {
        fieldName: "[Measures].[Internet Sales Amount]",
      }
    ],
    axis: "columns"
  }
],
filters: []
},
});
```

The above code will generate a simple PivotGrid with “Fiscal” field in Row, “Customer Geography” field in Column and “Internet Sales Amount” field in Value section.

|           | ► Australia           | ► Canada       | ► France       | ► Germany      | ► United Kingdom | ► United States | Total           |
|-----------|-----------------------|----------------|----------------|----------------|------------------|-----------------|-----------------|
|           | Internet Sales Amount |                |                |                |                  |                 |                 |
| ► FY 2002 | \$2,568,701.39        | \$573,100.97   | \$414,245.32   | \$513,353.17   | \$550,507.33     | \$2,452,176.07  | \$7,072,084.24  |
| ► FY 2003 | \$2,099,585.43        | \$305,010.69   | \$633,399.70   | \$593,247.24   | \$696,594.97     | \$1,434,296.26  | \$5,762,134.30  |
| ► FY 2004 | \$4,383,479.54        | \$1,088,879.50 | \$1,592,880.75 | \$1,784,107.09 | \$2,140,388.50   | \$5,483,882.67  | \$16,473,618.05 |
| ► FY 2005 | \$9,234.23            | \$10,853.70    | \$3,491.95     | \$3,604.83     | \$4,221.41       | \$19,434.51     | \$50,840.63     |
| Total     | \$9,061,000.58        | \$1,977,844.86 | \$2,644,017.71 | \$2,894,312.34 | \$3,391,712.21   | \$9,389,789.51  | \$29,358,677.22 |

## PivotGauge

### Overview

The **PivotGauge** control is a lightweight control that reads both **OLAP** and **Relational** datasources.

### Key Features

Some of the key features of PivotGauge are as follows,

- **Data source:** Supports OLAP data binding with **XML/A** data sources and Relational data sources.
- **Tooltip:** Displays the value and goal information in the tooltip.
- **Multiple gauges and Layouts:** Support to customize the layout while rendering multiple controls.
- **Frame types:** Built-in frame types provide a rich appearance of control.
- **Indicators:** Displays the active/inactive state of PivotGauge.
- **Ranges:** Highlighting the range of values in scale.
- **Pointers:** Points the actual value and goal information.

- **Header labels:** Support to show or hide header labels and indicators.

## Getting Started

This section explains you the steps required to populate the PivotGauge with data source. This section covers only the minimal features that you need to know to get started with the PivotGauge.

For common steps of typescript , you can refer [here](#).

The default type definition file ej.web.all.d.ts needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the jquery.d.ts file in your TypeScript application, which can be downloaded from [here](#).

## Script and CSS Reference

Add the scripts and CSS references to the "index.html" page as the order mentioned in the following code example.

### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
<script
src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.valida
te.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

In the above code, `ej.web.all.min.js` script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

## Relational

This section covers the information that you need to know to populate a simple PivotGauge with Relational data source.

### Control Initialization

Add a `div` container to render the PivotGauge.

#### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="PivotGauge1"></div>
</body>
</html>
```

Initialize the PivotGauge in app.ts file by using the `ej.PivotGauge` method.

#### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotGauge($("#PivotGauge1"), { });
});
```

### Populate PivotGauge with Data

Let us now see how to populate the PivotGauge control using a sample JSON data as shown below.

#### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
var pivot_dataset = [
{ Amount: 100, Country: "Canada", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Alberta" },
{ Amount: 200, Country: "Canada", Date: "FY 2006", Product: "Van",
Quantity: 3, State: "British Columbia" },
{ Amount: 300, Country: "Canada", Date: "FY 2007", Product: "Car",
Quantity: 4, State: "Brunswick" },
{ Amount: 150, Country: "Canada", Date: "FY 2008", Product: "Bike",
Quantity: 3, State: "Manitoba" },
{ Amount: 200, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 4, State: "Ontario" },
{ Amount: 100, Country: "Canada", Date: "FY 2007", Product: "Van",
Quantity: 1, State: "Quebec" },
{ Amount: 200, Country: "France", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Charente-Maritime" },
{ Amount: 250, Country: "France", Date: "FY 2006", Product: "Van",
Quantity: 4, State: "Essonne" },
{ Amount: 300, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 3, State: "Garonne (Haute)" },
{ Amount: 150, Country: "France", Date: "FY 2008", Product: "Van",
Quantity: 2, State: "Gers" },
{ Amount: 200, Country: "Germany", Date: "FY 2006", Product: "Van",
Quantity: 3, State: "Bayern" },
```

```

{ Amount: 250, Country: "Germany", Date: "FY 2007", Product: "Car",
Quantity: 3, State: "Brandenburg" },
{ Amount: 150, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 4, State: "Hamburg" },
{ Amount: 200, Country: "Germany", Date: "FY 2008", Product: "Bike",
Quantity: 4, State: "Hessen" },
{ Amount: 150, Country: "Germany", Date: "FY 2007", Product: "Van",
Quantity: 3, State: "Nordrhein-Westfalen" },
{ Amount: 100, Country: "Germany", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Saarland" },
{ Amount: 150, Country: "United Kingdom", Date: "FY 2008", Product:
"Bike", Quantity: 5, State: "England" },
{ Amount: 250, Country: "United States", Date: "FY 2007", Product: "Car",
Quantity: 4, State: "Alabama" },
{ Amount: 200, Country: "United States", Date: "FY 2005", Product: "Van",
Quantity: 4, State: "California" },
{ Amount: 100, Country: "United States", Date: "FY 2006", Product:
"Bike", Quantity: 2, State: "Colorado" },
{ Amount: 150, Country: "United States", Date: "FY 2008", Product: "Car",
Quantity: 3, State: "New Mexico" },
{ Amount: 200, Country: "United States", Date: "FY 2005", Product:
"Bike", Quantity: 4, State: "New York" },
{ Amount: 250, Country: "United States", Date: "FY 2008", Product: "Car",
Quantity: 3, State: "North Carolina" },
{ Amount: 300, Country: "United States", Date: "FY 2007", Product: "Van",
Quantity: 4, State: "South Carolina" }
];
$(function () {
var sample = new ej.PivotGauge($("#PivotGauge1"), {
dataSource: {
data: pivot_dataset,
rows: [
{
fieldName: "Country"
},
{
fieldName: "State"
}
],
columns: [
{
fieldName: "Product"
}
],
values: [
{
fieldName: "Amount",
fieldCaption: "Amount"
},
{
fieldName: "Quantity",
fieldCaption: "Quantity"
}
]
},
enableTooltip: true, isResponsive: true,
labelFormatSettings: { decimalPlaces: 2 },

```

```
scales: [{
  showRanges: true,
  radius: 150, showScaleBar: true, size: 1,
  border: {
    width: 0.5
  },
  showIndicators: true, showLabels: true,
  pointers: [{
    showBackNeedle: true,
    backNeedleLength: 20,
    length: 120,
    width: 7
  },
  {
    type: "marker",
    markerType: "diamond",
    distanceFromScale: 5,
    placement: "center",
    backgroundColor: "#29A4D9",
    length: 25,
    width: 15
  }],
  ticks: [{
    type: "major",
    distanceFromScale: 2,
    height: 16,
    width: 1, color: "#8c8c8c"
  },
  {
    type: "minor",
    height: 6,
    width: 1,
    distanceFromScale: 2,
    color: "#8c8c8c"
  }],
  labels: [{
    color: "#8c8c8c"
  }],
  ranges: [{
    distanceFromScale: -5,
    backgroundColor: "#fc0606",
    border: { color: "#fc0606" }
  },
  {
    distanceFromScale: -5
  }],
  customLabels: [{
    position: { x: 180, y: 290 },
    font: { size: "10px", fontFamily: "Segoe UI", fontStyle: "Normal" },
    color: "#666666"
  },
  {
    position: { x: 180, y: 320 },
    font: { size: "10px", fontFamily: "Segoe UI", fontStyle: "Normal" },
    color: "#666666"
  },
  {
```



```
position: { x: 180, y: 150 },
font: { size: "12px", fontFamily: "Segoe UI", fontStyle: "Normal" },
color: "#666666"
}]
}]
});
});
```

The above code will generate a simple PivotGauge as shown in below figure.



## OLAP

This section covers the information that you need to know to populate a simple PivotGauge with OLAP data source.

### Control Initialization

Add a `div` container to render the PivotGauge.

### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="PivotGauge1"></div>
</body>
</html>
```

Initialize the PivotGauge in ts file by using the `ej.PivotGauge` method.

### HTML

```
/// <reference path="jquery.d.ts" />
```

```
/// <reference path="ej.web.all.d.ts" />
$(function () {
  var sample = new ej.PivotGauge($("#PivotGauge1"), { });
});
```

### *Populate PivotGauge with data*

Let us now see how to populate the PivotGauge control using a sample JSON data as shown below.

### **HTML**

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
  var sample = new ej.PivotGauge($("#PivotGauge1"), {
    dataSource: {
      data: "http://bi.syncfusion.com/olap/msmdpump.dll",
      catalog: "Adventure Works DW 2008 SE",
      cube: "Adventure Works",
      rows: [
        {
          fieldName: "[Date].[Fiscal]",
          filterItems: { filterType: "include", values: ["[Date].[Fiscal].[Fiscal Year].&#38;[2004]"] }
        },
      ],
      columns: [
        {
          fieldName: "[Customer].[Customer Geography]"
        }
      ],
      values: [
        {
          measures: [
            {
              fieldName: "[Measures].[Internet Sales Amount]"
            },
            {
              fieldName: "[Measures].[Internet Revenue Status]"
            },
            {
              fieldName: "[Measures].[Internet Revenue Trend]"
            },
            {
              fieldName: "[Measures].[Internet Revenue Goal]"
            }
          ],
          axis: "columns"
        }
      ],
      filters: []
    },
    enableTooltip: true, isResponsive: true,
    labelFormatSettings: { decimalPlaces: 2 },
    scales: [{
      showRanges: true,
      radius: 150, showScaleBar: true, size: 1,
```

```
border: {
width: 0.5
},
showIndicators: true, showLabels: true,
pointers: [{
showBackNeedle: true,
backNeedleLength: 20,
length: 120,
width: 7
},
{
type: "marker",
markerType: "diamond",
distanceFromScale: 5,
placement: "center",
backgroundColor: "#29A4D9",
length: 25,
width: 15
}],
ticks: [{
type: "major",
distanceFromScale: 2,
height: 16,
width: 1, color: "#8c8c8c"
},
{
type: "minor",
height: 6,
width: 1,
distanceFromScale: 2,
color: "#8c8c8c"
}],
labels: [{
color: "#8c8c8c"
}],
ranges: [{
distanceFromScale: -5,
backgroundColor: "#fc0606",
border: { color: "#fc0606" }
},
{
distanceFromScale: -5
}],
customLabels: [{
position: { x: 180, y: 290 },
font: { size: "10px", fontFamily: "Segoe UI", fontStyle: "Normal" },
color: "#666666"
},
{
position: { x: 180, y: 320 },
font: { size: "10px", fontFamily: "Segoe UI", fontStyle: "Normal" },
color: "#666666"
},
{
position: { x: 180, y: 150 },
font: { size: "12px", fontFamily: "Segoe UI", fontStyle: "Normal" },
color: "#666666"
}
```

```
}]  
}]  
});  
});
```

The above code will generate a simple PivotGauge as shown in below figure.



## Frame type

### Full circle

The full circle frame is used to display the pivot gauge in circular shape. The frame type can be set by using the `frameType` property. By default, the frame type is "FullCircle".

### HTML

```
/// <reference path="jquery.d.ts" />  
/// <reference path="ej.web.all.d.ts" />  
$(function () {  
    var sample = new ej.PivotGauge($("#PivotGauge1"), {  
        frame: {  
            frameType: "fullcircle"  
        },  
    });  
});
```

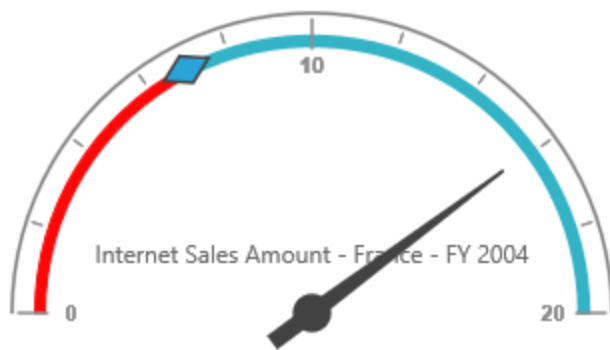


### Half circle

The half circle frame is used to display the pivot gauge in semi-circular shape. For this, set the frame type to "HalfCircle" within the `frameType` property, and then set the `startAngle` and `sweepAngle` for the pivot gauge in the `scales` property.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotGauge($("#PivotGauge1"), {
        frame: {
            frameType: "halfcircle",
            halfCircleFrameStartAngle: 180,
            halfCircleFrameEndAngle: 360
        },
        scales: [{
            startAngle: 180,
            sweepAngle: 180
        }]
    });
});
```



## Scale

### Adding scale

The scale can be added to the pivot gauge control.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
  var sample = new ej.PivotGauge($("#PivotGauge1"), {
    scales: [{
      radius: 150,
      showScaleBar: true
    }]
  });
});
```



## Scale customization

### Pointer cap

The pointer cap is a circular shape element located at the center of the pivot gauge. It can be customized with the `pointerCap` property in the scales option. Following are the properties used to customize its appearance.

- **radius:** Sets the radius of the pointer cap.
- **borderColor:** Sets the color of the pointer cap border.
- **borderWidth:** Sets the width of the pointer cap border.
- **backgroundColor:** Sets the background color of the pointer cap.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
  var sample = new ej.PivotGauge($("#PivotGauge1"), {
    scales: [{
      radius: 150,
      showScaleBar: true,
      pointerCap: {
        radius: 5,
        borderWidth: 2,
        borderColor: "green",
        backgroundColor: "yellow"
      }
    }
  ]
});
});
```



### Appearance

The appearance of the scale can be customized through the following properties:

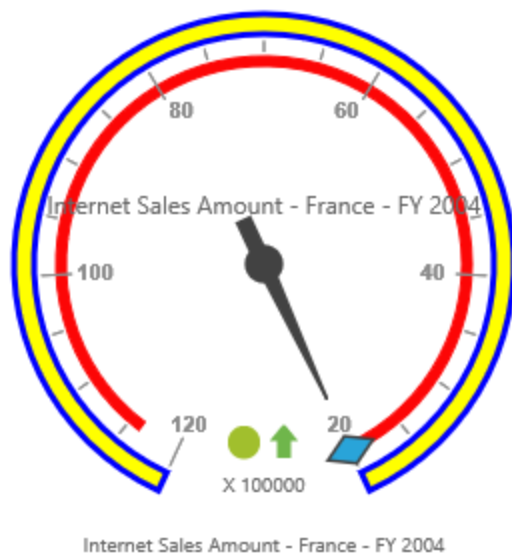
- **radius**: Sets the radius of the scale.
- **backgroundColor**: Sets the background color of the scale.
- **border**: Sets the border of the scale with color and width properties.
- **size**: Sets the size of the scale.
- **minimum**: Sets the least value of the scale.
- **maximum**: Sets the highest value of the scale.
- **majorIntervalValue**: Sets the interval between minor ticks in the scale.
- **minorIntervalValue**: Sets the interval between major ticks in the scale.
- **direction**: Sets the direction of the scale. By default, it takes clockwise direction.

The `showIndicators`, `showTicks`, `showRanges`, `showPointers`, and `showScaleBar` properties are used to enable/disable the indicators, ticks, ranges, pointers, and scale bar respectively. By default, the `showTicks` and `showPointers` are set to true, and other properties are set to false.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
  var sample = new ej.PivotGauge($("#PivotGauge1"), {
    scales: [{
      radius: 120, showScaleBar: true,
      size: 10,
      backgroundColor: "yellow",
      border: {
        width: 3,
        color: "blue"
      },
    },
    minimum: 20,
    maximum: 120,
    majorIntervalValue: 20,
    minorIntervalValue: 5,
    direction: "counterClockwise"
  }]
});
```





## Pointers

### Pointer types

The pivot gauge has two types of pointers namely,

- Needle
- Marker

Needle type pointer is the default pointer that is always located at the center of the gauge. Following are the supported pages for needle pointers:

- Rectangle
- Triangle
- Trapezoid
- Arrow
- Image

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotGauge($("#PivotGauge1"), {
scales: [{
pointers: [{
type: "needle",
needleType: "trapezoid"
}]
}]
});
});

```



For marker pointer, the available shapes are rectangle, triangle, ellipse, diamond, pentagon, circle, slider, pointer, wedge, trapezoid, rounded rectangle, and image.

#### **HTML**

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
  var sample = new ej.PivotGauge($("#PivotGauge1"), {
    scales: [{
      pointers: [{
        type: "marker",
        markerType: "diamond"
      }]
    }]
  });
});
```



### Adding pointer collection

The pointer collection can be directly added to the scales option within the pivot gauge control.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotGauge($("#PivotGauge1"), {
        scales: [{
            pointers: [{
                type: "needle",
                needleType: "triangle"
            },
            {
                type: "marker",
                markerType: "diamond"
            }
        ]
    }];
});
```



### Appearance Customization

The appearance of the pointer can be customized through the following properties:

- **border:** Sets the color and width of the pointer border.
- **backgroundColor:** Sets the background color of the pointer.
- **length:** Sets the length of the pointer.
- **width:** Sets the width of the pointer.
- **opacity:** Sets the opacity of the pointer. By default, the value is 1.
- **type:** Sets the type of the pointer. By default, the type is needle.

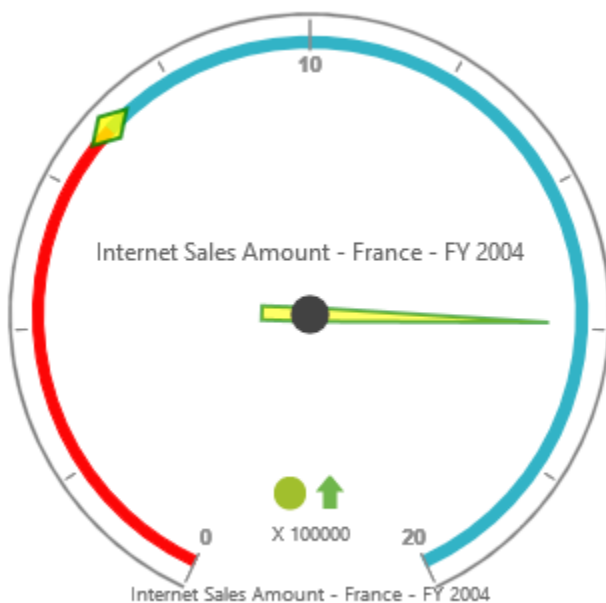
### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotGauge($("#PivotGauge1"), {
        scales: [{
            pointers: [{
                border:{
                    color: "green",
                    width:2
                },
                backgroundColor: "yellow",
                length:120,
                width:7,
                opacity:0.6,
                type: "needle",
                needleType: "triangle"
            },
            {
                border:{
                    color: "green",
```

```

width:2
},
backgroundColor: "yellow",
length:25,
width:15,
opacity:0.8,
type: "marker",
markerType: "diamond"
}]
}]
});
});

```



### Pointer position

The pointer can be positioned with the help of the following two properties:

- **distanceFromScale:** Defines the distance between the scale and the pointer. By default, the value is 0.
- **placement:** Defines the location of the pointer. By default, the value is center.

---

**Note:** Both the properties can be applied only if the pointer type is set to marker. The needle pointer type appears only at the center of the control, which is its default position.

---

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotGauge($("#PivotGauge1"), {
scales: [{
pointers: [
{
type: "marker",

```

```

distanceFromScale: 2,
placement: "far"
}]
}]
});
});

```



### Pointer image

You can replace the pointers with an image. To view the pointers as an image, set the appropriate location in the `imageUrl` property.

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotGauge($("#PivotGauge1"), {
scales: [{
pointers: [
{
type: "needle",
needleType: "image",
imageUrl: "../image.png"
},
{
type: "marker",
markerType: "image",
imageUrl: "../image.png"
}]
}]
});
});

```



### Pointer value text

To display the current value of pointers in the pivot gauge control, the **"pointerValueText"** option in pointers is used. Following are the properties used to enable and customize the pointer value text:

- **showValue:** Enables the pointer value text by setting the property to true. By default, its value is true.
- **distance:** Sets the distance between the pointer and the text.
- **color:** Sets the color of the text.
- **opacity:** Sets the opacity of the text. By default, its value is 1.
- **angle:** Sets the rotation angle of the text. By default, its value is 0.
- **font:** Sets the font size, font style, and font family of the text.

### HTML

```

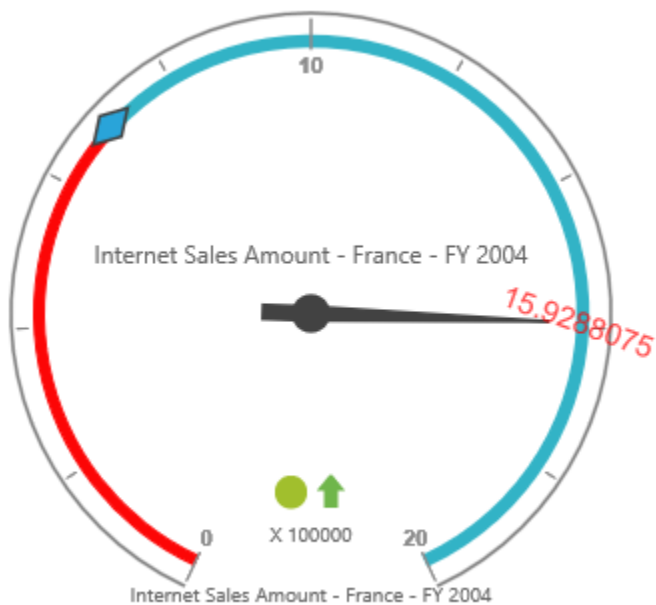
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotGauge($("#PivotGauge1"), {
scales: [{
pointers: [
{
type: "needle",
pointerValueText:{
showValue: true,
distance: 10,
color: "red",
opacity: 70,
angle: 20,
font:{
fontFamily: "Arial",
fontStyle: "Normal",
size: "15px"
}
}
}
]
}
}
);
});

```

```

}
}
},
{
  type: "marker",
  pointerValueText: {
    showValue: true,
    distance: 40,
    color: "red",
    opacity: 70,
    angle: -40,
    font: {
      fontFamily: "Arial",
      fontStyle: "Normal",
      size: "15px"
    }
  }
}
}
}
});
});

```



## Labels

### Adding label collection

Label collection can be directly added to the scales option within the pivot gauge control.

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
  var sample = new ej.PivotGauge($("#PivotGauge1"), {
    labels: [{

```



```
angle: 20
}],
});
});
```

### Appearance customization

The appearance of the label can be customized through the following properties:

- **angle**: Displays labels in a rotated manner. By default, the value is 0.
- **color**: Displays the label in a specified color.
- **opacity**: Sets the opacity of the label. By default, the value is 1.
- **type**: Indicates the label for major intervals or minor intervals. By default, it takes major intervals.
- **includeFirstValue**: Includes the initial value based on user requirement. By default, the value is true.
- **font**: Sets the font size, font style, and font family of the label.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotGauge($("#PivotGauge1"), {
labels: [{
color: "#1AFF01",
opacity: 80,
includeFirstValue: false,
font: {
size: "15px",
fontFamily: "Arial",
fontStyle: "Bold"
},
type: "major"
},
{
color: "#FF103F",
opacity: 80,
includeFirstValue: true,
font: {
size: "10px",
fontFamily: "Arial",
fontStyle: "Normal"
},
type: "minor"
}
],
});
});
```



### Unit text

The `unitText` property is used to add some text along with labels. Normally, the unit/measurement of the numeric value is indicated through the unit text. By using the `unitTextPosition` property, the text can be positioned in front or back.

**Note:** By default, text appears at the back.

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotGauge($("#PivotGauge1"), {
labels: [{
type: "major",
unitText: "$",
unitTextPosition: "front"
},
{
type: "minor",
unitText: "$",
unitTextPosition: "front"
}
],
});
});

```



## PivotTreeMap

### Overview

The **PivotTreeMap** control lets the user to visualize OLAP data in the form of nested nodes in hierarchical order with the ability to drill up and down.

### Key Features

Some of the key features of PivotTreeMap are as follows,

- **Data Source** - Binds the PivotTreeMap control with XML/A data sources.
- **Drill Support** - Enables you to navigate through the inner levels of a hierarchy elements.
- **Color Mapping** - Allows user to differentiate leaf nodes using various color codes either based on their value or members.
- **Legend** - Differentiates the color code based on value range (from minimum to maximum).

### Getting Started

This section explains you the steps required to populate the PivotTreeMap with data source. This section covers only the minimal features that you need to know to get started with the PivotTreeMap.

For common steps of typescript , you can refer [here](#).

The default type definition file ej.web.all.d.ts needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the jquery.d.ts file in your TypeScript application, which can be downloaded from [here](#).

### Script and CSS Reference

Add the scripts and CSS references to the "index.html" page as the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
<script
src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.valida
te.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

In the above code, ej.web.all.min.js script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

### Control Initialization

Add a **div** container to render the PivotTreeMap.

#### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="PivotTreeMap1" style="min-height: 275px; min-width: 525px;
height: 460px; width: 99%"></div>
<!--Tooltip labels can be localized here-->
<script id="tooltipTemplate" type="application/jsrender">
<div style="background:White; color:black; font-size:12px; font-
weight:normal; border: 1px solid #4D4D4D; white-space: nowrap;border-
radius: 2px; margin-right: 25px; min-width: 110px;padding-right: 5px;
padding-left: 5px; padding-bottom: 2px ;width: auto; height: auto;">
<div>Measure(s) : {{:~Measures(#data)}}</div><div>Row :
{{:~Row(#data)}}</div><div>Column : {{:~Column(#data)}}</div><div>Value :
{{:~Value(#data)}}</div>
</div>
</script>
</body>
```

```
</html>
```

Initialize the PivotTreeMap in ts file by using the `ej.PivotTreeMap` method.

### HTML

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
  var sample = new ej.PivotTreeMap($("#PivotTreeMap1"), { });
});
```

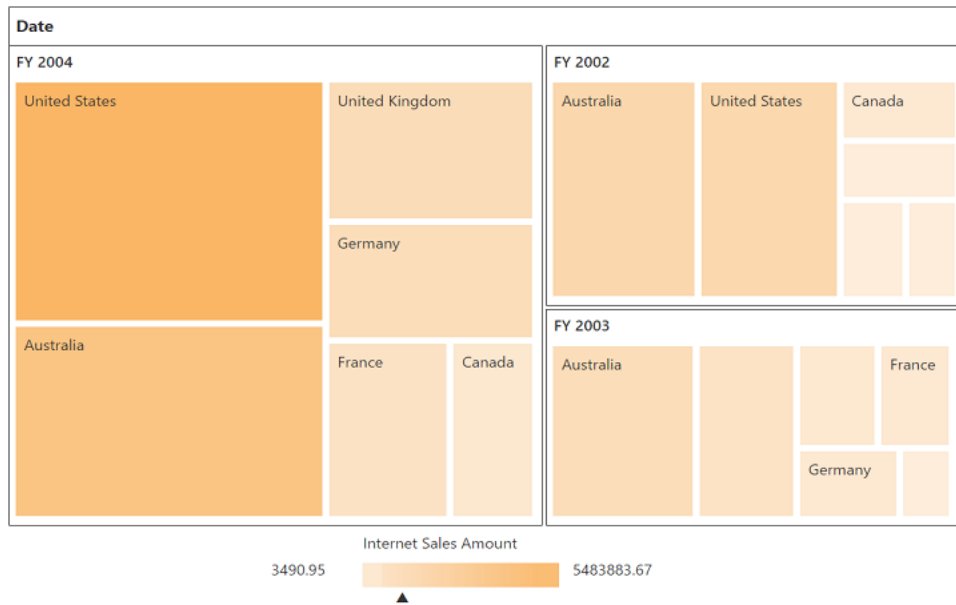
### Populate PivotTreeMap with data

Let us now see how to populate the PivotTreeMap control using a sample JSON data as shown below.

### HTML

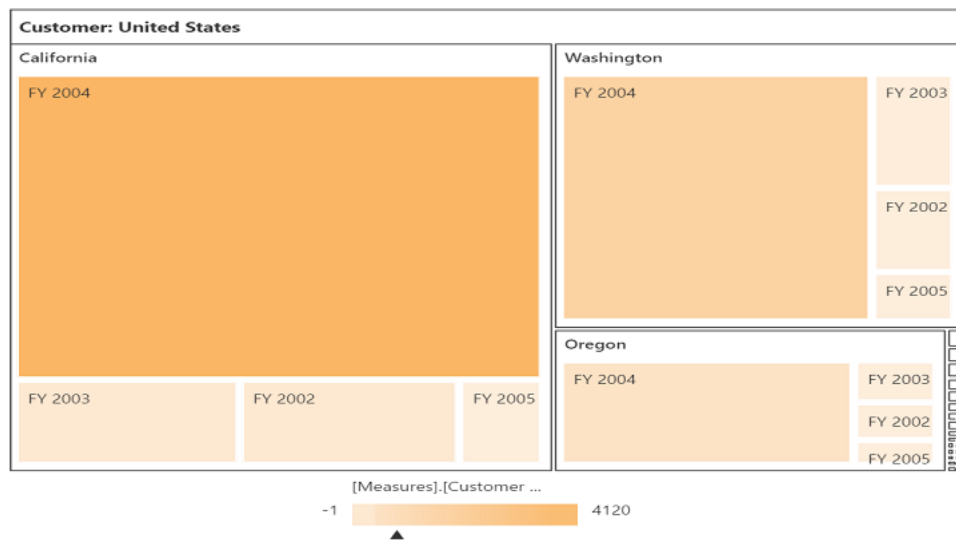
```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
  var sample = new ej.PivotTreeMap($("#PivotTreeMap1"), {
    dataSource: {
      data: "http://bi.syncfusion.com/olap/msmdpump.dll;Locale
Identifier=1033;",
      catalog: "Adventure Works DW 2008 SE",
      cube: "Adventure Works",
      rows: [
        {
          fieldName: "[Date].[Fiscal]"
        }
      ],
      columns: [
        {
          fieldName: "[Customer].[Customer Geography]"
        }
      ],
      values: [
        {
          measures: [
            {
              fieldName: "[Measures].[Customer Count]",
            }
          ],
          axis: "columns"
        }
      ],
      filters: []
    }
  });
});
```

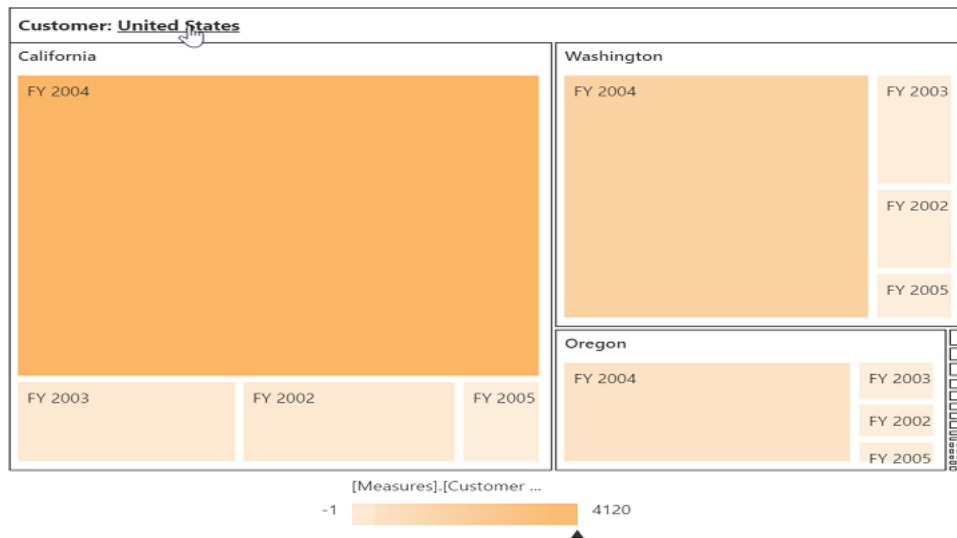
The above code will generate a simple PivotTreeMap with customer count over a period of fiscal years across different customer geographic locations.



### Drill operation

Drill down exposes the hierarchy by clicking a node; this results in allowing the tree map to move to the next level or sub level. You can return back to the normal tree map view by clicking the node header at the top.





## Named sets

Named sets is a multidimensional expression (MDX) that returns a set of dimension members that can be created by combining cube data, arithmetic operators, numbers, and functions.

You can bind the named sets in the pivot tree map by setting its unique name in the `fieldName` property in the row or column axis and `isNamedSets` Boolean property to true.

## HTML

```

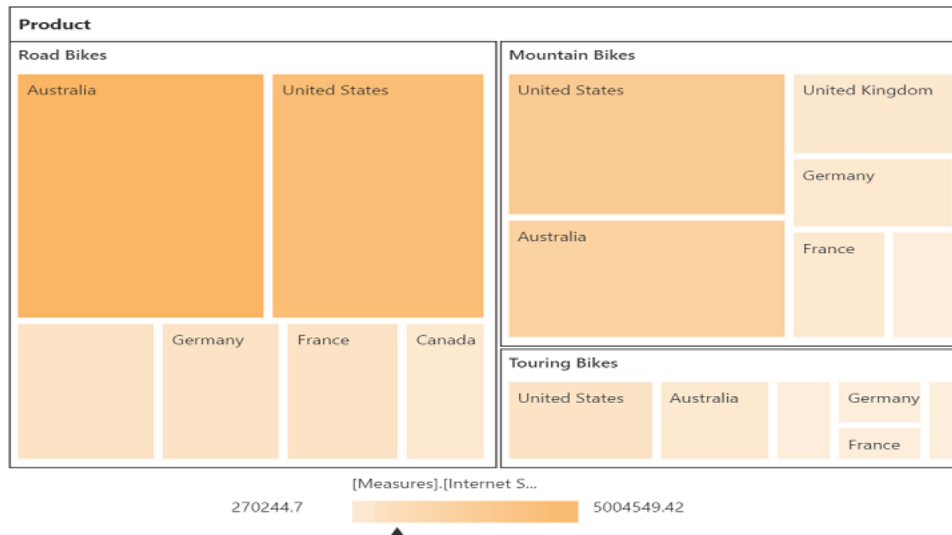
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotTreeMap($("#PivotTreeMap1"), {
        dataSource: {
            data: "http://bi.syncfusion.com/olap/msmdpump.dll;Locale
Identifier=1033;",
            catalog: "Adventure Works DW 2008 SE",
            cube: "Adventure Works",
            rows: [
                {
                    fieldName: "[Core Product Group]",
                    isNamedSets: true
                }
            ],
            columns: [
                {
                    fieldName: "[Customer].[Customer Geography]"
                }
            ],
            values: [
                {
                    measures: [
                        {
                            fieldName: "[Measures].[Customer Count]",
                        }
                    ]
                }
            ],
            axis: "columns"
        }
    });

```

```

],
filters: []
}
});
});

```



## Color mapping

You can customize the colors of leaf nodes of the pivot tree map using the color mapping support.

Color mapping is categorized into two different types as follows:

- Normal
- Range

You can color all the leaf nodes with different colors by setting the `color` value of the `rangeColorMapping` property.

### Normal color mapping

You can customize the nodes based on number of leaf items using different color ranges. You can also define the color value range using `from` and `to` properties.

The following code sample explains how to customize the pivot tree map appearance using **Normal** mode (i.e., differentiate color based on number of leaf items in each header).

### HTML

```

<!DOCTYPE html>
<html>
<body>
<div id="PivotTreeMap1" style="min-height: 275px; min-width: 525px;
height: 460px; width: 99%"></div>
<!--Tooltip labels can be localized here-->
<script id="tooltipTemplate" type="application/jsrender">
<div style="background:White; color:black; font-size:12px; font-
weight:normal; border: 1px solid #4D4D4D; white-space: nowrap;border-
radius: 2px; margin-right: 25px; min-width: 110px;padding-right: 5px;
padding-left: 5px; padding-bottom: 2px ;width: auto; height: auto;">

```



```

<div>Measure(s) : {{:~Measures(#data)}}</div><div>Row :
{{:~Row(#data)}}</div><div>Column : {{:~Column(#data)}}</div><div>Value :
{{:~Value(#data)}}</div>
</div>
</script>
</body>
</html>

```

## HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
var sample = new ej.PivotTreeMap($("#PivotTreeMap1"), {
renderSuccess: function (args) {
let treemapTarget =
$('#PivotTreeMap1TreeMapContainer').data("ejTreeMap");
treemapTarget.model.colorValuePath = "";
treemapTarget.model.enableGradient = false;
treemapTarget.model.showLegend = false;
treemapTarget.model.legendSettings.leftLabel = "";
treemapTarget.model.legendSettings.rightLabel = "";
treemapTarget.model.rangeColorMapping = [];
treemapTarget.model.colorValuePath = "Index";
treemapTarget.model.rangeColorMapping.push(
{ color: "#9de24f", from: 0, to: 0 },
{ color: "#a2e2fe", from: 1, to: 1 },
{ color: "#ffff66", from: 2, to: 2 },
{ color: "#FF0040", from: 3, to: 3 },
{ color: "#f6b53f", from: 4, to: 4 },
{ color: "#6FAAB0", from: 5, to: 5 },
{ color: "#C4C24A", from: 6, to: 6 }
)
treemapTarget.refresh();
}
});
});

```



*Range color mapping*

You can customize the nodes based on its value and color ranges using **Range** color. You can also define the color value range by using **from** and **to** properties.

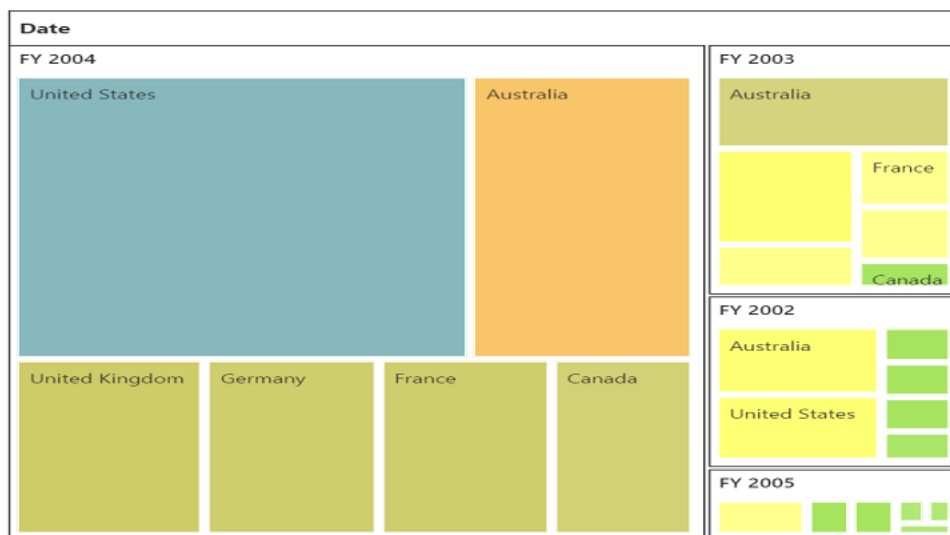
The following code sample explains how to customize the pivot tree map appearance using **Range** mode (i.e., differentiate color for leaf items based on values).

**HTML**

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotTreeMap($("#PivotTreeMap1"), {
        renderSuccess: function (args) {
            let treemapTarget =
                $('#PivotTreeMap1TreeMapContainer').data("ejTreeMap");
            treemapTarget.model.colorValuePath = "";
            treemapTarget.model.enableGradient = false;
            treemapTarget.model.showLegend = false;
            treemapTarget.model.legendSettings.leftLabel = "";
            treemapTarget.model.legendSettings.rightLabel = "";
            treemapTarget.model.rangeColorMapping = [];
            treemapTarget.model.colorValuePath = "Value";
            treemapTarget.model.rangeColorMapping.push(
                { color: "#9de24f", from: 0, to: 10 },
                { color: "#a2e2fe", from: 11, to: 250 },
                { color: "#ffff66", from: 251, to: 1000 },
                { color: "#FF0040", from: 1001, to: 3000 },
                { color: "#f6b53f", from: 3001, to: 5000 },
                { color: "#6FAAB0", from: 5001, to: 10000 },
                { color: "#C4C24A", from: 10001, to: 20000 }
            )
            treemapTarget.refresh();
        }
    });
});

```

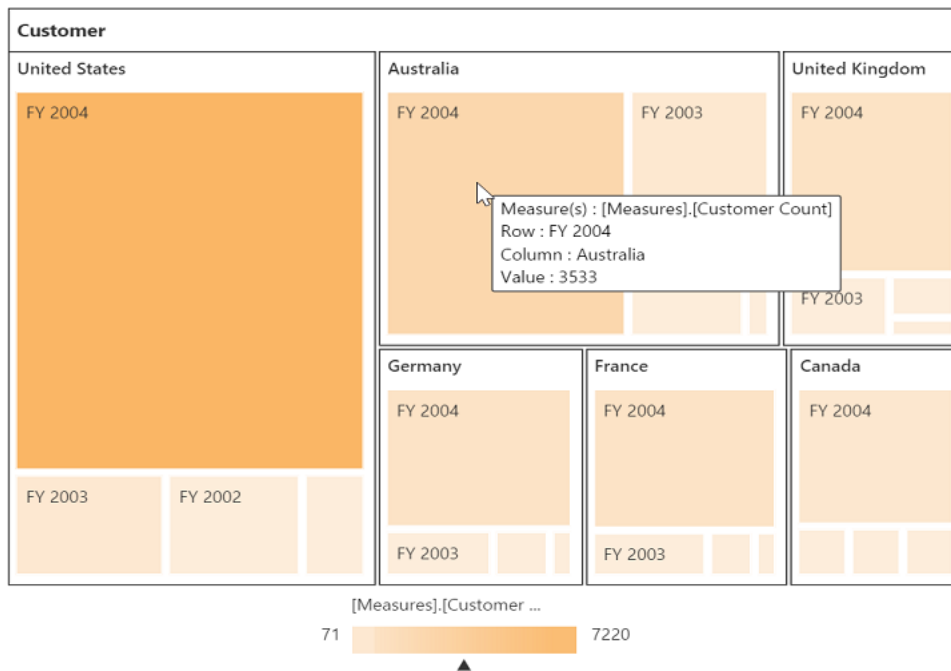


## Legend

### Legend visibility

The legend shows the value range differences and color occurrence in the respective leaf node while hovering it with the cursor.

**Note:** By default, the legend is visible in the pivot tree map.



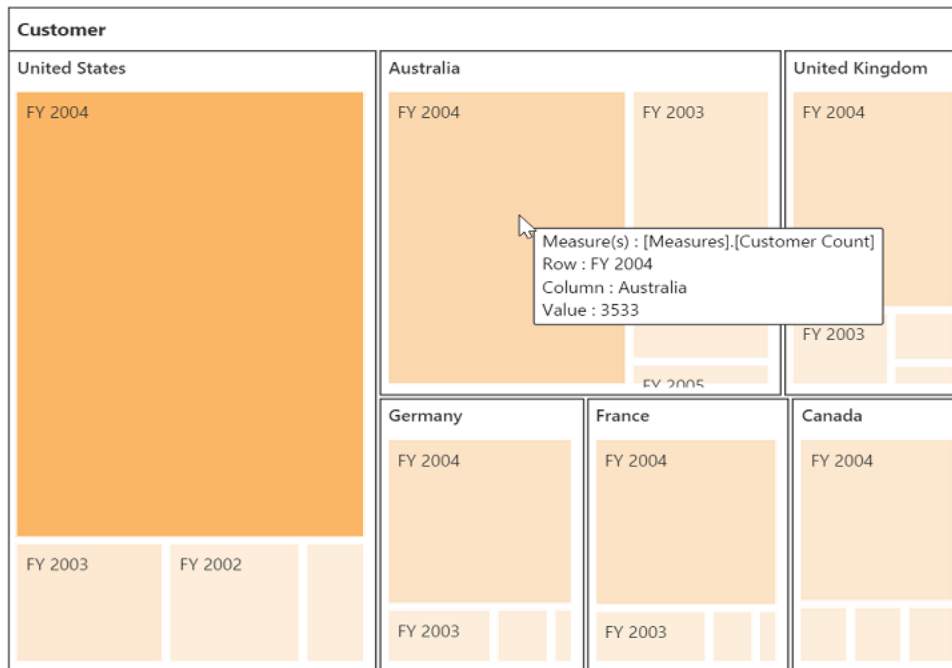
You can disable the legend by setting the **showLegend** property to **false**. The following code example shows how to disable the legend:

### HTML

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
$(function () {
    var sample = new ej.PivotTreeMap($("#PivotTreeMap1"), {
        renderSuccess: function (args) {
            let treemapTarget =
                $('#PivotTreeMap1TreeMapContainer').data("ejTreeMap");
            treemapTarget.model.showLegend = false;
            treemapTarget.refresh();
        }
    });
});

```



## ProgressBar

### Overview

**Essential TypeScript ProgressBar** control is a graphical control element

<http://en.wikipedia.org/wiki/Graphicalcontrol/element> used to visualize the changing status of an extended operation. It is available in **JavaScript** product. **ProgressBar** provides an interactive way to display the progression of the task. You can configure the item size, orientation and the display text on the **ProgressBar** control.

### Key Features

Some important features of **ProgressBar** are as follows:

- **Value support** — you can set a numeric value for the **ProgressBar** status.
- **Percentage support** — you can set a percentage value for the control.
- **CustomText** — **ProgressBar** control allows you to set the text value that is to be displayed inside the control.
- **StateMaintenance** — the progress value is maintained even after the page is refreshed to resume the current progress of the operation.
- **Range** — to set the minimum and maximum values of the **ProgressBar** control
- **Theme** — **ProgressBar** control consists of 17 built-in themes (6 flat, 6 gradient, bootstrap, 2 high-contrast, material and office-365 effects) and also support the custom skin option to set user-defined themes.

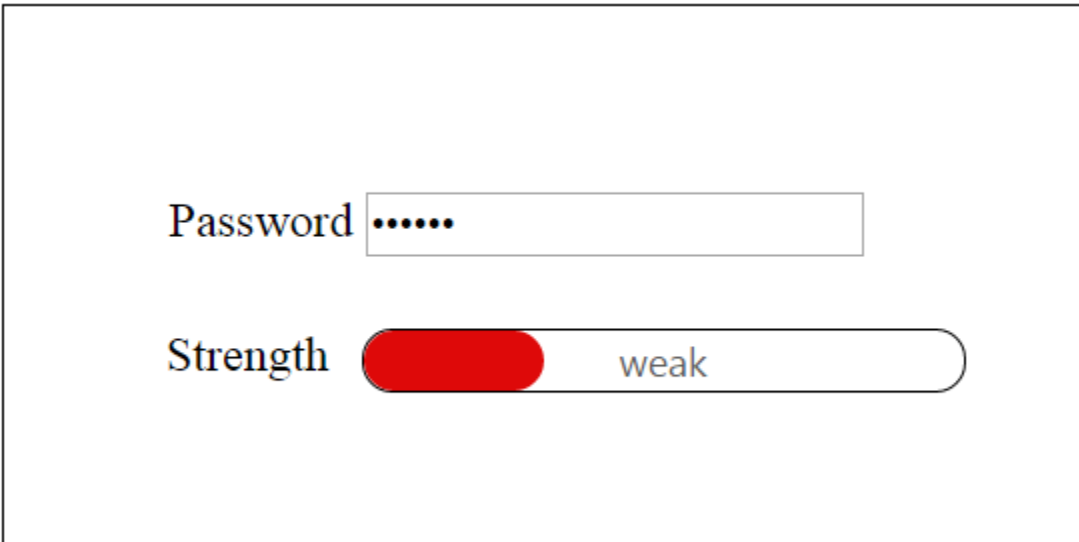
### Getting Started

This section briefly describes how to create a **ProgressBar** control using **TypeScript** and learn its features.

**Essential TypeScript ProgressBar** displays a **ProgressBar** within a web page that allows you to show the progress of an event. Here, you can learn how to customize the progress and color of the **ProgressBar** in

a real-time application to indicate the strength of the password, where the progress changes with respect to the change in length of the password. This helps you to validate the password is typed.

The following screenshot shows the **ProgressBar**.



#### Create a ProgressBar

**Essential TypeScript ProgressBar** widget is created using a simple **<div>** element. This element provides built-in features that allow you to change the progress, size and text of the control.

You can create the **ProgressBar** widget by using a simple input **<div>** element as follows:

Create an **HTML** file and add the following template to the **HTML** file to create your ProgressBar. It also includes the necessary scripts and styles.

#### HTML

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>Getting Started - RichTextEditor</title>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"></script>
</head>
<body>
<!-- add rating element here -->
</body>
</html>
```

Add **<input>** element inside the **<body>** tag of your file to create a **ProgressBar**.

#### HTML

```
<div style="content-container-fluid">
```

```

<div class="row">
<div class="cols-sample-area">
<div class="frame">
<div class="wrap_up">
<!--Initializing password field-->
<label for="startButton">Password</label>
<input type="password" id="password" style="border-radius:0px"/>
</div>
<div class="control">
<!--initializing ProgressBar control-->
<div id="progressBar"></div>
</div>
</div>
</div>
</div>
</div>

```

It also includes a Password field and through that the progress of the **ProgressBar** can be controlled

Initialize the ProgressBar in ts file by using the `ej.ProgressBar` method.

#### JAVASCRIPT

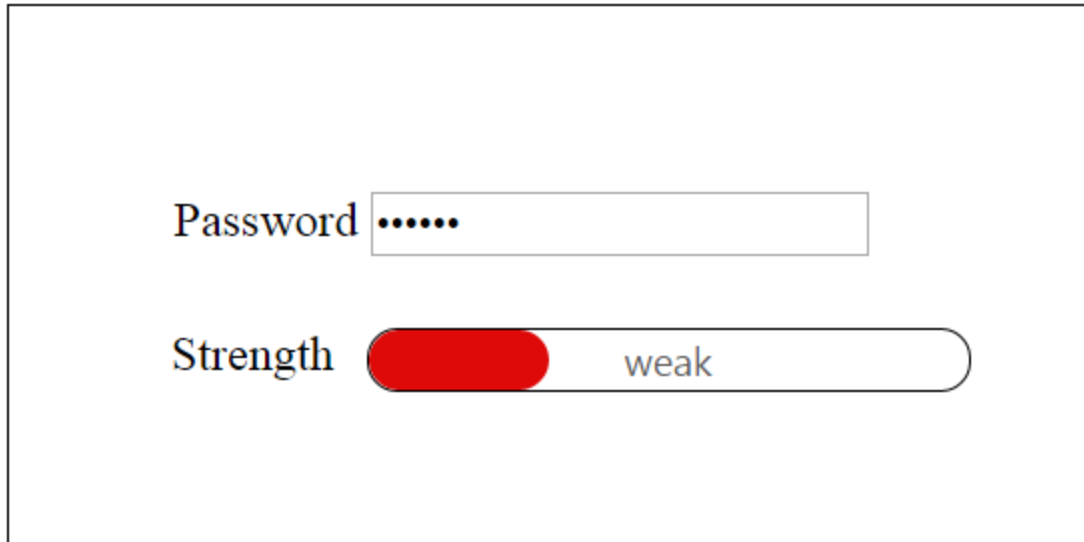
```

/// <reference path="jquery.d.ts" />
/// <reference path="scripts/typings/ej/ej.web.all.d.ts" />
module ProgressBarComponent {
$(function () {
var progress = new ej.ProgressBar($("#progressBar"), {
height: 20,
value: 30, /*Specify the initial value of the progress in percentage*/
width: 200,
});
progress.option("text", "weak");
$(".e-progress").css({ "background-color": "#DE0909", "border-radius":
"10px" });
$(".e-progressbar").css({ "border-radius": "10px", "border": "1px solid
black" });
});
}

```

Here, you can initialize the properties of the **ProgressBar** such as height, value, width, text that is applied to the control by default.

The following screenshot displays a **ProgressBar** control.



Include the following code within the **<head>** tag to change the page layout.

### CSS

```
<style type="text/css" class="cssStyles">  
/*applying styles */  
.frame {  
border: 1px solid #BBBCBB;  
border-radius: 10px 10px 10px 10px;  
padding: 50px 60px;  
margin-top: 40px;  
width: 400px;  
margin-left: 400px;  
}  
.control {  
margin-bottom: 5px;  
margin-left: 230px;  
}  
.wrap_up {  
margin-left: 105px;  
font-size: 18px;  
}  
#progressBar {  
margin-top: 10px;  
}  
</style>
```

### Progress Control using Length of the Password Field

In real-time scenario, the progress of **ProgressBar** is changed according to the length of text in the password field by binding the change in the properties of control and checking the length of the password field.

Add the following code example inside the **<script>** tag of your **HTML** file.

### JAVASCRIPT

```
///<reference path="jquery.d.ts" />
```

```

/// <reference path="scripts/typings/ej/ej.web.all.d.ts" />
module ProgressBarComponent {
var progresObj, buttonObj, k = 10, timer = window.clearInterval(timer), i
= 0, obj;
$(function () {
var progresObj = new ej.ProgressBar($("#progressBar"), {
height: 20,
value: 30, /*Specify the initial value of the progress in percentage*/
width: 200,
});
progresObj.option("text", "weak");
$(".e-progress").css({ "background-color": "#DE0909", "border-radius":
"10px" });
$(".e-progressbar").css({ "border-radius": "10px", "border": "1px solid
black" });
});
$(document).keypress(function () { //To capture the keypress inside
the document
i = $("#password").val().length;
if (i < 5)
weak();
else if (i == 5 || i < 7)
Strong();
else if (i == 7 || i > 7)
very_strong();
});
function Strong() { //Change the width and text of the progress ...
called when the length is greater than 5
progresObj.option("text", "strong");
progresObj.option("percentage", k + 50);
$(".e-progress").css("background-color", "#0055FF");
$(".e-progressbar").css("color", "#000000");
}
function very_strong() { //Change the width and text of the progress
... called when the length is greater than 7
progresObj.option("text", "Very strong");
progresObj.option("percentage", k + 90);
$(".e-progress").css("background-color", "Green");
$(".e-progressbar").css("color", "#000000");
}
function weak() { //Change the width and text of the progress...
called when the length is less than 5
progresObj.option("text", "Weak");
progresObj.option("percentage", k + 20);
$(".e-progress").css("background-color", "#DE0909");
$(".e-progressbar").css("border-radius", "10px");
}
}

```

You can calculate length of the password and call the appropriate function that changes the percentage property of **ProgressBar**.

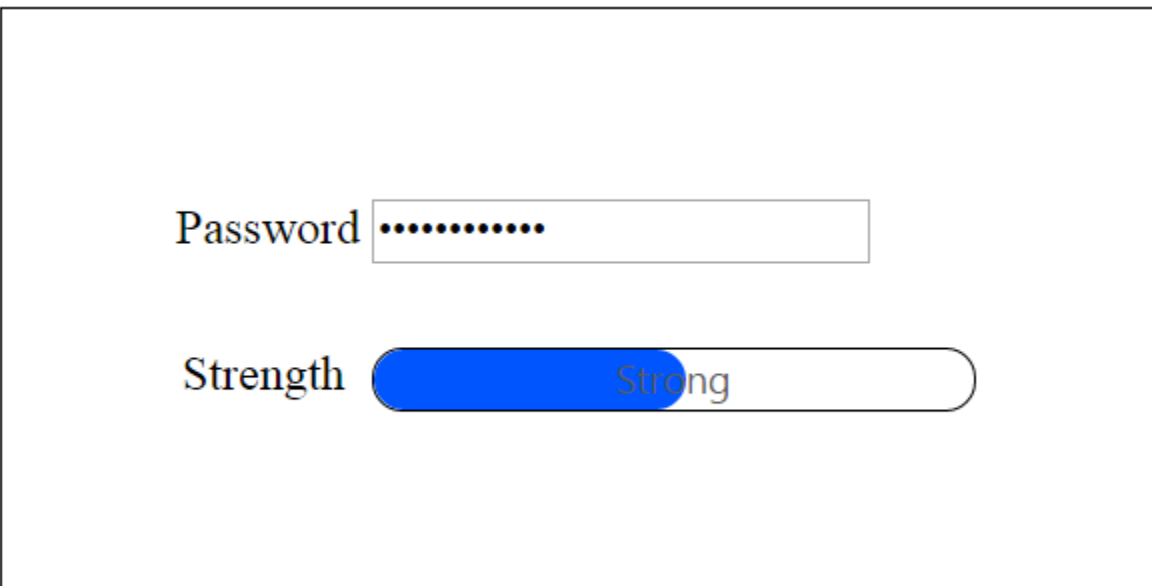
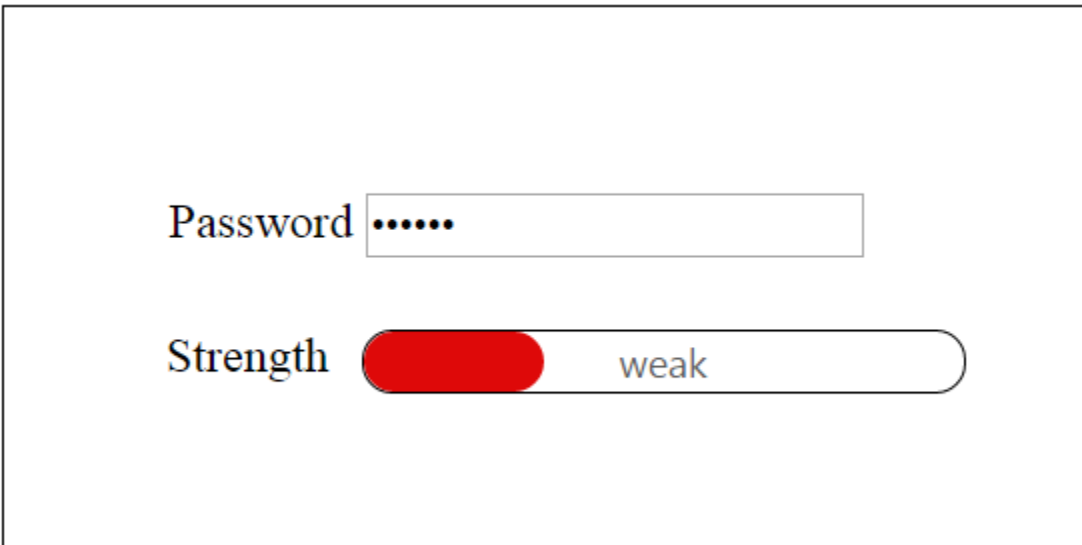
- The **weak()** function changes the text inside the ProgressBar to **Weak** and percentage to 30, that is invoked when the length of the text is less than 5.

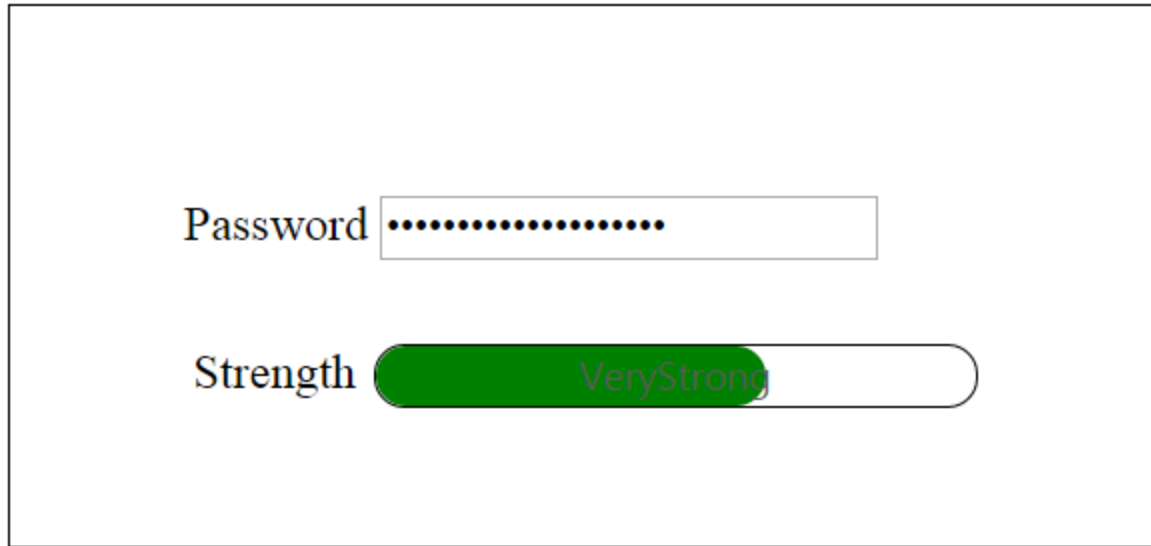


- The **strong()** function changes the text inside the ProgressBar to **Strong** and percentage to 60, that is invoked when the length of the text exceeds 5.
- The **very\_strong()** function changes the text inside the ProgressBar to **Very Strong** and percentage to 100, that is invoked when the length of the text exceeds 7 and the text contains a symbol in it.

You can change themes or appearance of the ProgressBar as required.

The final output is displayed as follows.





You can also bind an event at the start and finish of a ProgressBar by using the start, complete and change properties of the ProgressBar.

### Define value

#### Value

The value for the ProgressBar is set by using '**value**' property. The value should be between the minimum (min) and the maximum (max) values (number) of the ProgressBar. By default, the **minValue** is **0** and the **maxValue** is **100** in ProgressBar, and the '**value**' is set to **0**(number).

The following steps explain you on how to set the **value** for the ProgressBar widget.

In the **HTML** page, add a **<div>** element to render the ProgressBar widget.

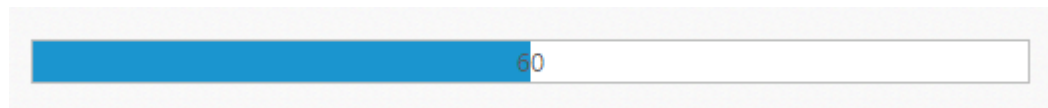
#### HTML

```
<div class="control">
  <div id="progressbar"></div>
</div>
```

#### JAVASCRIPT

```
// Add the following code example to set the value for the ProgressBar
widget.
module ProgressBarComponent {
  $(function () {
    var sample = new ej.ProgressBar($("#progressbar"), {
      minValue: 40,
      maxValue: 80,
      value: 60,
      height: 20,
      width: 500
    });
    progress.sample({ text: progress.getValue() });
  });
}
```

The following screenshot displays the output for the above code.



### Percentage

The ProgressBar value is set in ProgressBar by using the '**percentage**' property. The value should be between the min and max values (number) of the ProgressBar. By default, the **minValue** is **0** and the **maxValue** is **100** in ProgressBar, and **percentage** is set to **0** (number).

The following steps explain you on how to set the value in **percentage** for the ProgressBar widget.

In the **HTML** page, add a **<div>** element to render the ProgressBar widget.

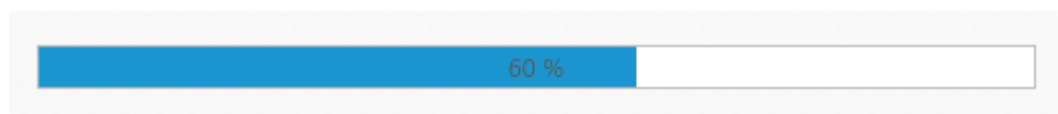
### HTML

```
<div class="control">
<div id="progressbar"></div>
</div>
```

### JAVASCRIPT

```
// Add the following code example to set the value in percentage for the
//ProgressBar widget.
$(function () {
//Declaration.
var sample = new ej.ProgressBar($("#progressbar"), {
percentage: 60,
width: 500,
height: 20
});
sample.setModel({ text: progress.getValue() + " %" });
});
```

The following screenshot displays the output.



### Setting Range

The **range** of the ProgressBar is set by using minimum and maximum values. The Minimum value specifies the value where the ProgressBar shows the process to have started. The Maximum value specifies the value where the process is completed. You can set the range by using the '**minValue**' and '**maxValue**' property.

In the **HTML** page, add a **<div>** element to render the ProgressBar widget.

### HTML

```
<div class="control">
<div id="progressbar"></div>
</div>
```

**JS**

```
// Add Range for the ProgressBar widget as follows.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ProgressBarComponent {
  $(function () {
    var sample = new ej.ProgressBar($("#progressbar"), {
      minValue: 40,
      maxValue: 80,
      value: 80,
      height: 20,
      width: 500
    });
    sample.setModel({ text: progress.getPercentage() + " % Completed" });
  });
}
```

The following screenshot displays the output.

**Appearance and Styling****Adjusting ProgressBar size**

**ProgressBar** widget provides the ability to change or adjust the ProgressBar size. The **'height'** and **'width'** properties in the ProgressBar widget allows you to set the maximum height and maximum width for the ProgressBar. The value set to this property is **string or Number** type.

The following steps explain you on how to adjust the ProgressBar size.

In the **HTML** page, add a **<div>** element to render the **ProgressBar** widget

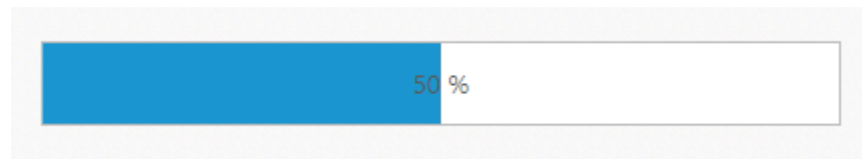
**HTML**

```
<div class="control">
<div id="progressbar"></div>
</div>
```

**JAVASCRIPT**

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ProgressBarComponent {
  $(function () {
    var sample = new ej.ProgressBar($("#progressbar"), {
      value: 50,
      width: "400px",
      height: "40px"
    });
    sample.setModel({ text: progress.getValue() + " %" });
  });
}
```

The following screenshot displays the output.



### Custom text

**Custom text** is displayed when the **ProgressBar** shows different levels of progress in the **ProgressBar**. Support for **Custom text** to mention the percentage or any other message inside the **ProgressBar** is possible by using **"text"** property.

The following steps explain the configuration of the **Custom text** for the **ProgressBar** widget.

In the **HTML** page, add a **<div>** element to render the **ProgressBar** widget.

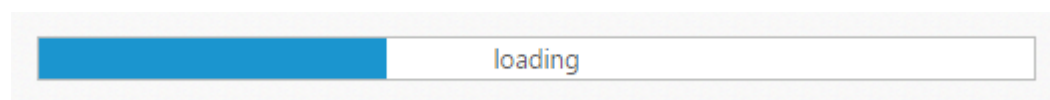
### HTML

```
<div class="control">
<div id="progressbar"></div>
</div>
```

### JAVASCRIPT

```
// Add Custom Text for the ProgressBar widget.
$(function () {
var sample = new ej.ProgressBar($("#progressbar"), {
text: "loading",
value: 35,
height: 20,
width: 500
});
});
```

The following screenshot displays the output.



### Theme

The **ProgressBar** widget style and appearance are controlled based on **CSS** classes. In order to apply **Theme** to the **ProgressBar** widget, you can refer two files, namely, **ej.widgets.core.min.css** and **ej.theme.min.css**. When the file **ej.widgets.all.min.css** is referred, then it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.widgets.all.min.css** is the combination of these both.

By default, there are 12 themes' support available for the **ProgressBar** widget namely,

- Default-theme
- Flat-azure-dark
- Fat-lime
- Flat-lime-dark
- Flat-saffron

- Flat-saffron-dark
- Gradient-azure
- Gradient-azure-dark
- Gradient-lime
- Gradient-lime-dark
- Gradient-saffron
- Gradient-saffron-dark

### CSS class

To apply custom styles to the ProgressBar widget, you can specify the **cssClass** property. The specified **CSS** name is added in the root of the **ProgressBar** widget.

The following code example is used to render the ProgressBar widget with customized style.

In the **HTML** page, add a **<div>** element to render the ProgressBar widget.

### HTML

```
<div class="control">
<div id="progressbar"></div>
</div>
```

### JAVASCRIPT

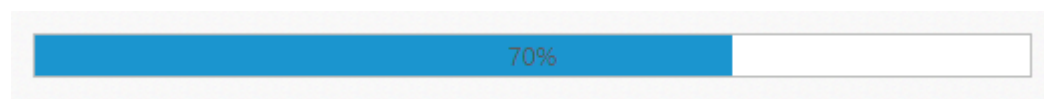
```
// Add the class for the ProgressBar Widget.
$(function () {
var sample = new ej.ProgressBar($("#progressbar"), {
cssClass: 'custom',
value: 70,
height: 20,
width: 500
});
progress.sample({ text: progress.getValue() + "%" });
});
```

Add the following styles to render the ProgressBar with customized style.

### CSS

```
<style type="text/css">
.custom .e-progress {
background-color:gray;
}
</style>
```

The following screenshot displays the output.



### Enabling the ProgressBar

The ProgressBar is enabled by using the **'enabled'** Property. When this property is set to **'false'**, it disables the ProgressBar widget. By default, **'enabled'** property is set to **'true'** in the ProgressBar widget.

The following steps explain how to disable the ProgressBar widget when 'enabled' property is set to 'false'.

In the **HTML** page, add a **<div>** element to render the ProgressBar widget.

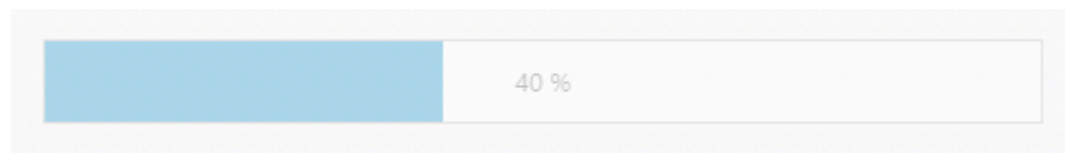
### HTML

```
<div class="control">
<div id="progressbar"></div>
</div>
```

### JAVASCRIPT

```
// Add the 'enabled' property as follows.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ProgressBarComponent {
$(function () {
var sample = new ej.ProgressBar($("#progressbar"), {
enabled: false,
value: 40,
width: 500,
height: 40
});
sample.setModel({ text: progress.getValue() + " %" });
});
}
```

The following screenshot displays the output for the above code.



### State Maintenance

Save current model value to cookies for **State Maintenance**. While refreshing the ProgressBar widget, page retains the model value applied from browser cookies. By default, the 'enablePersistence' property is set to 'false' in the ProgressBar.

The following steps explain the **State Maintenance** in the ProgressBar control.

In the **HTML** page, add a **<div>** element to render the ProgressBar widget.

### HTML

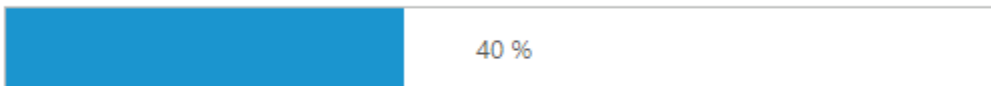
```
<div class="control">
<div id="progressbar"></div>
</div>
```

### JAVASCRIPT

```
// Add the following script to enable State Maintenance.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ProgressBarComponent {
```

```
$(function () {
  var sample = new ej.ProgressBar($("#progressbar"), {
    enablePersistence: true,
    value: 40,
    width: 500,
    height: 40
  });
  sample.setModel({ text: progress.getValue() + " %" });
});
```

The following screenshot displays the output.



## RTL Support

Right-to-left starts from the right of the page and continues to the left. By default, this option is set to 'false' in the ProgressBar control. The **enableRTL** option allows the ProgressBar control to display it in the right to left direction.

The following steps explain how to **enable** the **RTL** property of the ProgressBar control.

In the **HTML** page, add a **<div>** element to render the ProgressBar widget.

### HTML

```
<div class="control">
  <div id="progressbar"></div>
</div>
```

### JAVASCRIPT

```
// Add the following code example to enable the RTL property of the
// ProgressBar control.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ProgressBarComponent {
  $(function () {
    var sample = new ej.ProgressBar($("#progressbar"), {
      enableRTL: true,
      value: 80,
      text: 80,
      height: 20,
      width: 500
    });
    sample.setModel({ text: progress.getValue() + " % Completed" });
  });
}
```

The following screenshot displays the output.





## How To

How to increment & show the progressbar movement

You can increment the progress percentage and show case the movement by using the below code

### HTML

```
<div class="frame">
<div class="control">
<div id="progressBar"></div>
</div>
<div class="startButton">
<input type="checkbox" id="startButton" />
<label for="startButton">Toggle</label>
</div>
</div>
<div class="cols-prop-area event-tracer">
<div class="heading">
<span>Event Trace</span>
<div class="pull-right">
<select name="selectevtprops" id="selectControls">
<option value="start">start</option>
<option value="complete">complete</option>
<option value="change">change</option>
</select>
</div>
</div>
<div class="prop-grid content">
<div class="eventarea">
<div class="EventLog" id="EventLog">
</div>
</div>
<div class="evtbtn">
<input type="button" class="eventclear e-btn" value="Clear"
onclick="onClear()" />
</div>
</div>
</div>
```

### JAVASCRIPT

```
var , k = 10, timer = window.clearInterval(timer), showComplete=true ;
/// <reference path="../tsfiles/jquery.d.ts" />
/// <reference path="../tsfiles/ej.web.all.d.ts" />
module ProgressBarComponent {
$(function () {
var sample = new ej.ProgressBar($("#progressBar"), {
height: 22,
value: 10,
start: "onstart",
change: "onchange",
complete: "completed"
});
```

```

sample.option("text", progresObj.getPercentage() + " %");
var sample1 = new ej.ToggleButton($("#startButton"), {
  defaultText: "Start",
  activeText: "Pause",
  size: "small",
  click: "startProcess"
});
var sample2 = new ej.DropDownList($("#selectControls"), {
  popupShown: "adjustpopupposition",
  showCheckbox: true,
  checkAll: true,
  change: "evtpropscheckedevent"
});
});
function evtpropscheckedevent(args) {
  if (args.isChecked) {
    switch (args.value) {
      case "start": sample.option(args.value, "onstart"); break;
      case "change": sample.option(args.value, "onchange"); break;
      case "complete": showComplete=true; break;
    }
  }
  else if (args.value=="complete")
  {
    showComplete=false;
  }
  else
  sample.option(args.value, null)
}
function startProcess(args) {
  if (args.isChecked)
    timer = window.setInterval(draw, 100);
  else {
    sample1.setModel({ "defaultText": "Start" });
    timer = window.clearInterval(timer);
  }
}
function draw() {
  sample.option("text", ++k + " %");
  sample.option("percentage", k);
}
function completed(args) {
  sample.option("text", "Completed");
  timer = window.clearInterval(timer);
  k = 0;
  if (showComplete)
    oncomplete(args);
  sample1.setModel({ "toggleState": false, "defaultText": "Restart" });
}
function oncomplete(args) {
  jQuery.addEventLog("The process has been <span  
class='eventTitle'>completed</span> successfully.</br>");
}
function onstart(args) {
  jQuery.addEventLog("Progressbar has been <span  
class='eventTitle'>started</span>.</br>");
}
}

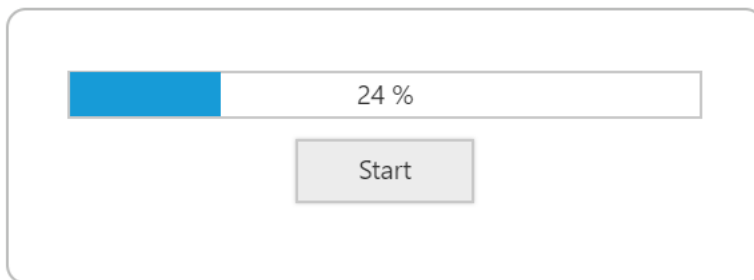
```

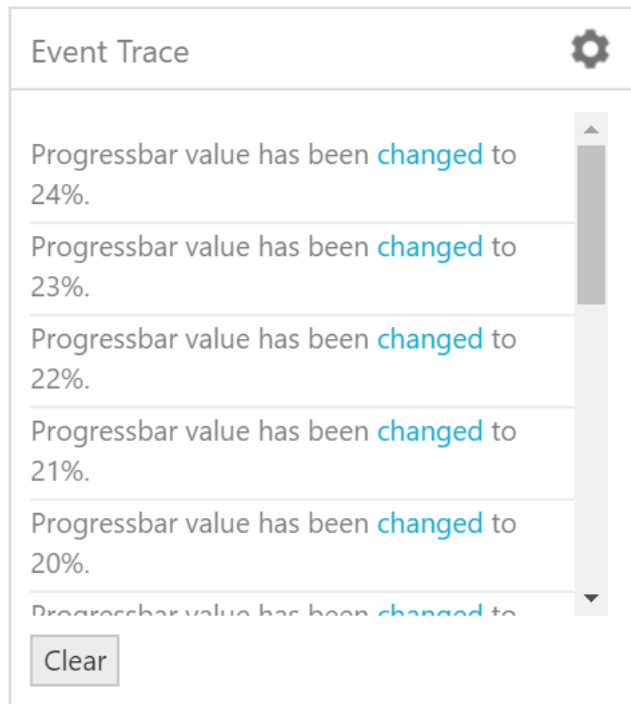
```
function onchange(args) {
jQuery.addEventLog("Progressbar value has been <span
class='eventTitle'>changed</span> to " + args.percentage + "%.</br>");
}
function onClear() {
$("#EventLog").html("");
}
```

## CSS

```
<style type="text/css" class="cssStyles">
.frame {
border: 1px solid #BBBCBB;
border-radius: 10px 10px 10px 10px;
padding: 50px 60px;
margin-top: 40px;
width: 400px;
}
.txt {
font-size: 20px;
margin-top: 12px;
text-align: center;
}
.startButton {
margin-left: 155px;
}
</style>
```

The progress bar movement will be as shown below:





## RadialMenu

### Overview

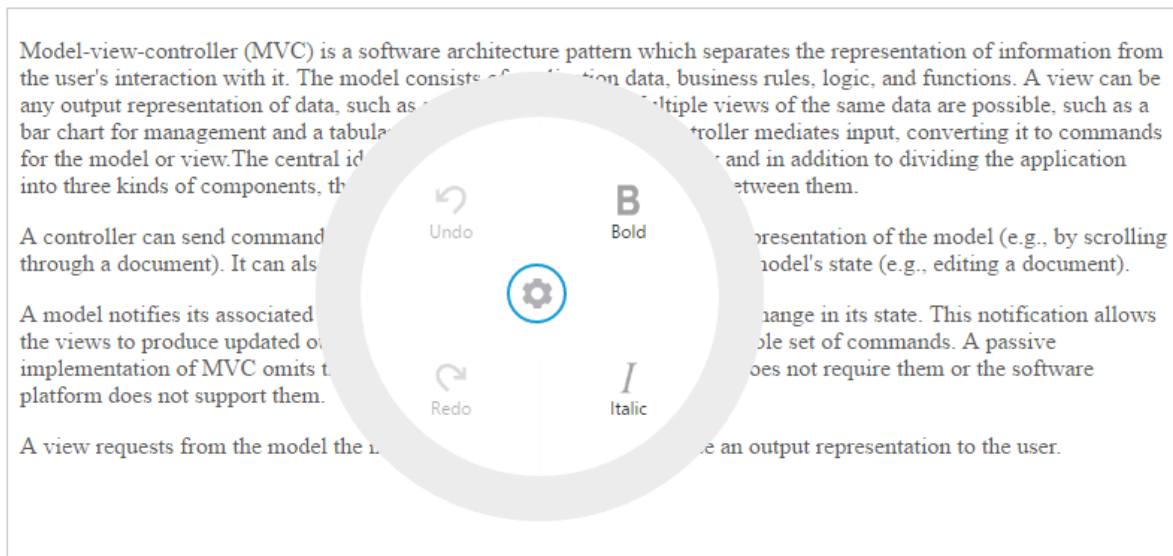
The Essential Typescript RadialMenu that represents the menu items are arranged in a circular order with a centric button element in it. By default, center button only is visible. The RadialMenu displays the root level menu item with rotational animation effects on clicking the center menu button. You can close it either by clicking anywhere in the document or by clicking the center button where the root level items are displayed.

### Key Features

- **Nested Menu:** Supports to render the multiple levels of sub-menu items.
- **Image Customization:** Enables to customize images for all levels of the menu item.
- **Item Customization:** Supports to customize RadialMenu items with badges and slider settings.
- **Dimension:** Allows to customize RadialMenu radius and position.

### Getting Started

Using the following steps, you can create a Typescript **RadialMenu** control. The basic rendering of Typescript RadialMenu is achieved with default functionality.



## Create a RadialMenu

Refer the common Typescript [Getting Started Documentation](#) to create an application and add necessary scripts and styles for rendering the Splitter control.

Create an HTML file and add the scripts references in the order mentioned in the following code example.

## HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Typescript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/fl
at-azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.
web.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add RadialMenu here-->
</body>
</html>
```

Create div element and add in the body tag. To create the RadialMenu, you should call the `ejRadialMenu` jQuery plug-in function.

## HTML

```
<div id="defaultRadialMenu">
<ul>
<li data-ej-imageurl="content/images/RadialMenu/font.png" data-ej-
text="Bold" data-ej-click="bold"></li>
<li data-ej-imageurl="content/images/RadialMenu/fl.png" data-ej-
text="Italic" data-ej-click="italic"></li>
```

```
<li data-ej-imageurl="content/images/RadialMenu/redo.png" data-ej-
text="Redo" data-ej-click="redo" data-ej-enabled="false"></li>
<li data-ej-imageurl="content/images/RadialMenu/undo.png" data-ej-
text="Undo" data-ej-click="undo" data-ej-enabled="false"></li>
</ul>
</div>
```

Initialize the RadialMenu in app.ts file by using the ej.RadialMenu method.

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module RadialMenuComponent {
$(function () {
let sample = new ej.RadialMenu($("#defaultRadialMenu"));
});
}
```

Now build your application, so that the app.js file is automatically generated and got added to your project (User have nothing to do with this file). Now, whatever code changes that you make in app.ts file will be reflected in app.js file automatically.

### Image and text configuration

You can set the images for each item by giving the image URL with the **data-ej-imageUrl** attribute in the inner list element and text with **data-ej-text** attribute. Refer to the following code example.

### HTML

```
<div id="defaultRadialMenu">
<ul>
<li data-ej-imageurl="content/images/RadialMenu/font.png" data-ej-
text="Bold" ></li>
<li data-ej-imageurl="content/images/RadialMenu/fl.png" data-ej-
text="Italic" ></li>
<li data-ej-imageurl="content/images/RadialMenu/redo.png" data-ej-
text="Redo" data-ej-enabled="false"></li>
<li data-ej-imageurl="content/images/RadialMenu/undo.png" data-ej-
text="Undo" data-ej-enabled="false"></li>
</ul>
</div>
```

Refer to the following code example to add target content to the **RadialMenu**.

### HTML

```
<div id="radialtarget1" class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<textarea id="rteSample1" rows="10" cols="70" style="height: 440px">
<p>
Model-view-controller (MVC) is a software architecture pattern which
separates the representation of information from the user's interaction
with it.
```

The model consists of application data, business rules, logic, and functions. A view can be any output representation of data, such as a chart or a diagram. Multiple views of the same data are possible, such as a bar chart for management and a tabular view for accountants. The controller mediates input, converting it to commands for the model or view. The central ideas behind MVC are code reusable and in addition to dividing the application into three kinds of components, the MVC design defines the interactions between them.

</p>

<p>A controller can send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document). It can also send commands to the model to update the model's state (e.g., editing a document).</p>

<p>A model notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. A passive implementation of MVC omits these notifications, because the application does not require them or the software platform does not support them.</p>

<p>A view requests from the model the information that it needs to generate an output representation to the user.</p>

</textarea>

</div>

</div>

</div>

Add the following styles in your code.

### CSS

```
.e-radialmenu .imageClass {
background-image: url(content/images/RadialMenu/settings.png);
}
.e-radialmenu .backImageClass {
background-image: url(content/images/RadialMenu/Back_button.png);
}
textarea {
padding: 10px;
}
#defaultRadialMenu {
pointer-events:none;
}
.e-radial,#defaultRadialMenu_svgdiv {
pointer-events:all;
}
```

### Displaying RadialMenu

You can display the Radial Menu by performing desired action on the target content while selecting the text inside the target. Therefore, call the **select** event of RTE the content. Refer to the following code example and add it to the script.

### JAVASCRIPT

```
declare var rteObj: any;
```

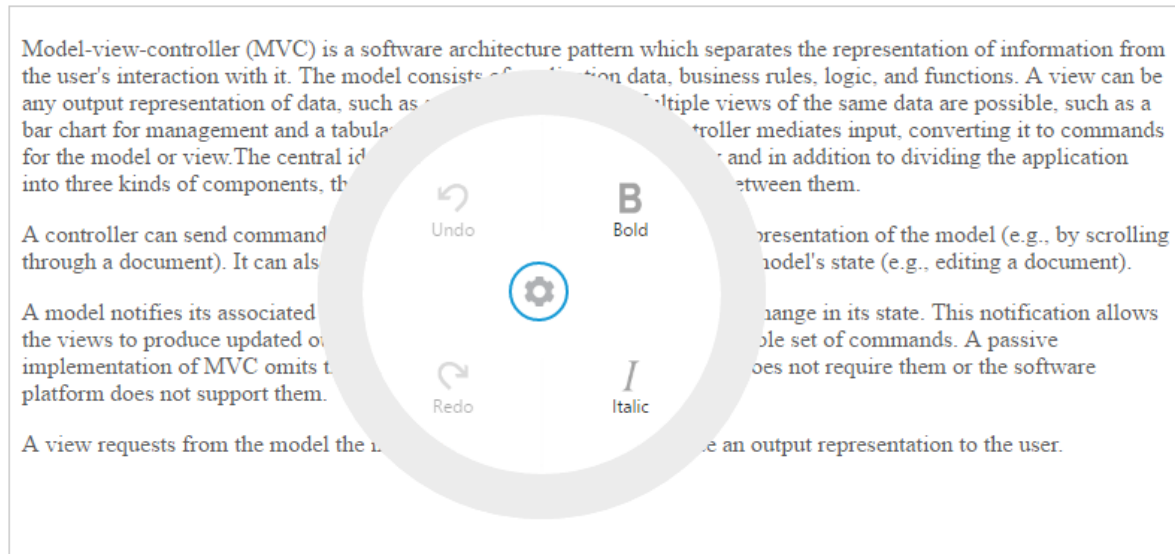
```

declare var data: any;
var radialElement = $('#defaultRadialMenu'), action = 0, forRedo = 0;
var rteElement = $("#rteSample1");
module RadialMenuComponent {
$(function () {
if (!(ej.browserInfo().name == "msie" &&
parseInt(ej.browserInfo().version) < 9)) {
var radialMenuInstance = new ej.RadialMenu($("#defaultRadialMenu"), {
imageClass: "imageClass",
backImageClass: "backImageClass",
targetElementId: "radialtarget1"
});
$("#radialtarget1").parent().css("position", "relative");
}
var rteInstance = new ej.RTE($("#rteSample1"), {
width: "100%",
minWidth: "10px",
change: (e) => { radialElement.ejRadialMenu("enableItem", "Undo"); },
select: (e) => {
var target = $("#radialtarget1"), radialRadius = 150, radialDiameter = 2
* radialRadius,
// To get Iframe positions
iframeY = e.event.clientY, iframeX = e.event.clientX,
// To set Radial Menu position within target
x = iframeX > target.width() - radialRadius ? target.width() -
radialDiameter : (iframeX > radialRadius ? iframeX - radialRadius : 0),
y = iframeY > target.height() - radialRadius ? target.height() -
radialDiameter : (iframeY > radialRadius ? iframeY - radialRadius : 0);
radialElement.ejRadialMenu("setPosition", x, y);
radialElement.focus();
$('iframe').contents().find('body').blur();
},
showToolbar: false,
showContextMenu: false
});
});
}

```

Run the above code and select any text inside the target. The settings icon is displayed. Click that icon to render the following output.





## RadialMenu item functionalities

You can set the functionalities for each item and define click function by using **data-ej-click** attribute. Refer to the following code example. Define the click function for Radial Menu control items as follows.

## HTML

```
<div id="defaultRadialMenu">
<ul>
<li data-ej-imageurl="content/images/RadialMenu/font.png" data-ej-
text="Bold" data-ej-click="bold"></li>
<li data-ej-imageurl="content/images/RadialMenu/fl.png" data-ej-
text="Italic" data-ej-click="italic"></li>
<li data-ej-imageurl="content/images/RadialMenu/redo.png" data-ej-
text="Redo" data-ej-click="redo" data-ej-enabled="false"></li>
<li data-ej-imageurl="content/images/RadialMenu/undo.png" data-ej-
text="Undo" data-ej-click="undo" data-ej-enabled="false"></li>
</ul>
</div>
```

Refer to the following code example to add functionalities for each items in **click** event and you can enable items in RadialMenu by using **change** event in typescript file.

# JAVASCRIPT

```
function bold(e: any) {
    rteObj = rteElement.data("ejRTE");
    rteObj.executeCommand("bold");
    data = rteObj._getSelectedHtmlString() ? true : false;
    if (data) action += 1;
    forRedo = action;
    radialElement.focus();
}

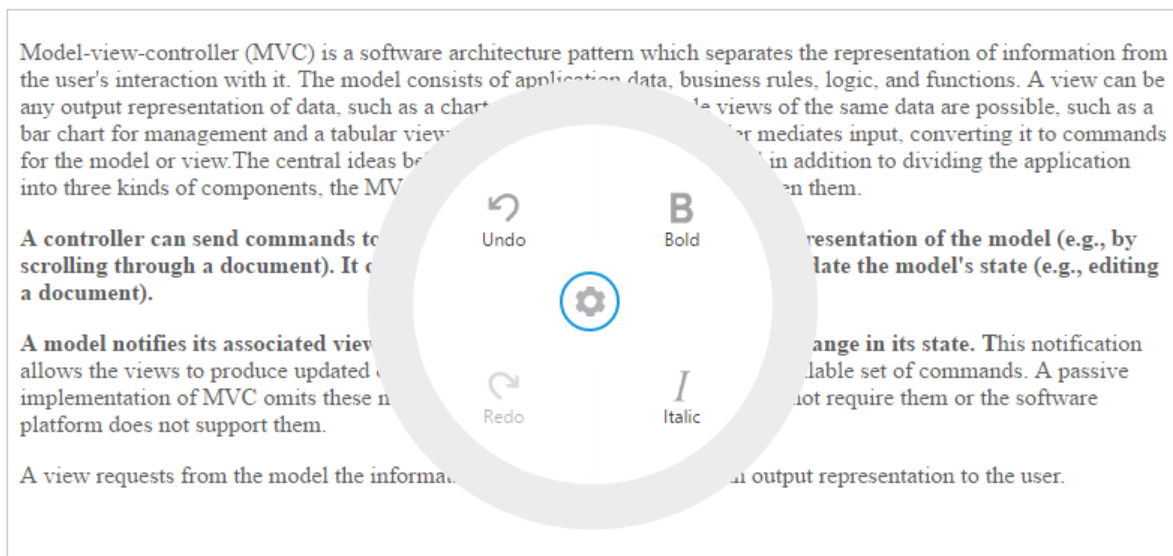
function italic(e: any) {
    rteObj = rteElement.data("ejRTE");
    rteObj.executeCommand("italic");
    data = rteObj._getSelectedHtmlString() ? true : false;
    if (data) action += 1;
}
```

```

forRedo = action;
radialElement.focus();
}
function undo(e: any) {
rteObj = rteElement.data("ejRTE");
rteObj.executeCommand("undo");
action -= 1;
if (action == 0)
radialElement.ejRadialMenu("disableItem", "Undo");
radialElement.ejRadialMenu("enableItem", "Redo");
radialElement.focus();
}
function redo(e: any) {
rteObj = rteElement.data("ejRTE");
rteObj.executeCommand("redo");
action += 1;
if (forRedo == action) radialElement.ejRadialMenu("disableItem", "Redo");
radialElement.ejRadialMenu("enableItem", "Undo");
radialElement.focus();
}

```

Run the above code and select any text inside the target. The settings icon is displayed. Click that icon to render the RadialMenu control. Click **bold** item in RadialMenu control, to render the following output.



**Note:-> You can find the RadialMenu properties from the [API reference](#) document**

## Dimension

You can customize **Radial Menu** dimension by using **radius** and **position** properties.

### Radius

You can customize the **Radial Menu** size by using the **radius** property. By default, the **Radial Menu** radius is set as 150px. Refer to the following code example.

## HTML

```

<!-- RTE code for setting target for Radial menu -->
<div id="defaulttradialmenu">

```

```

<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/font.png" data-
ej-text="Bold"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/fl.png" data-ej-
text="Italic"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/align.png" data-
ej-text="Align"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/sort.png" data-
ej-text="Sort"></li>
</ul>
</div>

```

Add the following script in your code.

### JAVASCRIPT

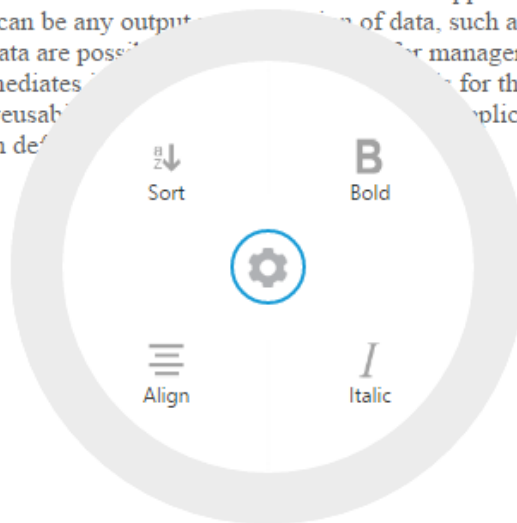
```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RadialMenuComponent {
$(function () {
var radialmenuInstance = new ej.RadialMenu($("#defaultradialmenu"), {
radius: "200"
});
});
$("#rteSampleone").select(function (e) {
$('#defaultradialmenu').ejRadialMenu("show");
});
}

```

The following screenshot illustrates the **Radial Menu** while clicking on the settings icon.

Model-view-controller (MVC) is a software architecture pattern which separates the representation of information from the user's interaction with it. The model consists of application data, business rules, logic, and functions. A view can be any output representation of data, such as a chart or a diagram. Multiple views of the same data are possible. For example, a tree view for management and a tabular view for accountants. The controller mediates between the user and the model or view. The central ideas behind MVC are code reusability and separation of concerns. An application is divided into three kinds of components, the MVC design defines the interaction between them.



### Position

To display the **Radial Menu** in the web page in a specific area, we can use the **position** property. By default, the **Radial Menu** position is set as null.

Refer to the following code example.

### HTML

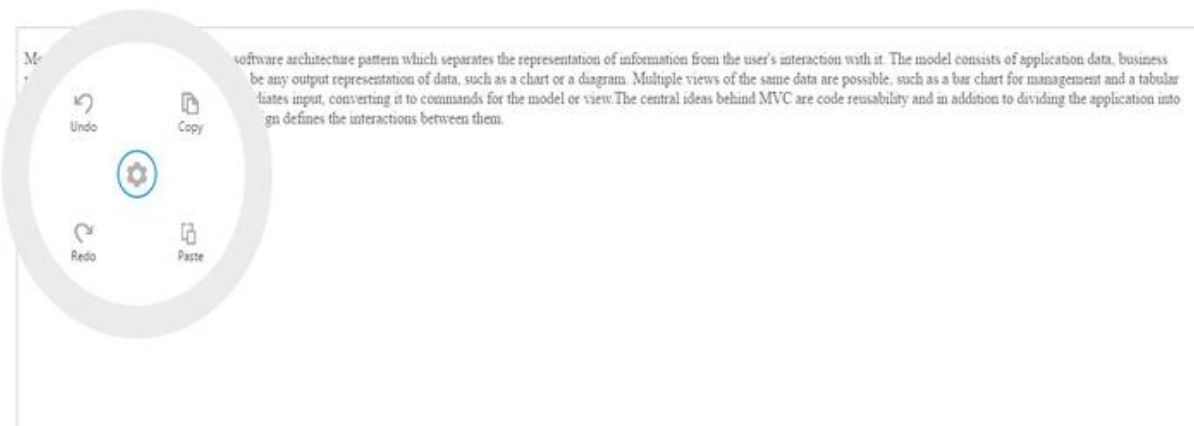
```
<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/copy.png" data-
ej-text="Copy"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/paste.png" data-
ej-text="Paste"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/redo.png" data-
ej-text="Redo"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/undo.png" data-
ej-text="Undo"></li>
</ul>
```

Add the following script in your code.

### JAVASCRIPT

```
$(function () {
var radialmenuInstance = new ej.RadialMenu($("#defaultradialmenu"), {
radius: "200",
position:{x:10,y:10}
});
});
$("#rteSampleone").select(function (e) {
$('#defaultradialmenu').ejRadialMenu("show");
});
```

The following screenshot illustrates the output while selecting the text in the page.



### Item Customization

You can customize individual **Radial Menu** items by using the items properties.

### *Adding image and text to RadialMenu items*

The **data-ej-imageUrl** property specifies the URL of the image for the items. **data-ej-text** attribute is used to specify the item text. Refer to the following code example.

#### **HTML**

```
<div id="defaulttradialmenu">
  <ul>
    <li data-ej-
      imageUrl="http://js.syncfusion.com/UG/web/Content/radial/copy.png" data-
      ej-text="Copy"></li>
    <li data-ej-
      imageUrl="http://js.syncfusion.com/UG/web/Content/radial/paste.png" data-
      ej-text="Paste"></li>
    <li data-ej-
      imageUrl="http://js.syncfusion.com/UG/web/Content/radial/redo.png" data-
      ej-text="Redo"></li>
    <li data-ej-
      imageUrl="http://js.syncfusion.com/UG/web/Content/radial/undo.png" data-
      ej-text="Undo"></li>
  </ul>
</div>
```

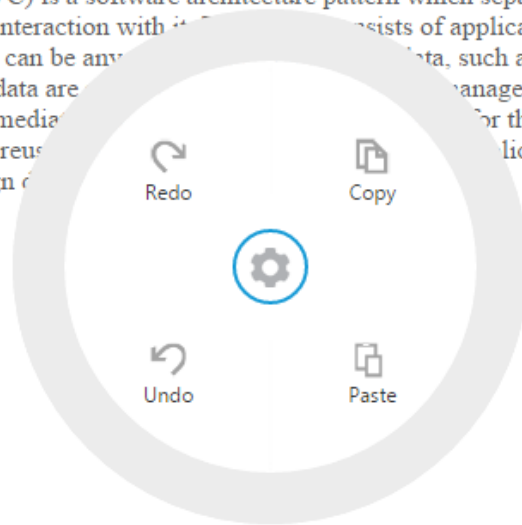
Add the following script in your code.

#### **JAVASCRIPT**

```
$(function () {
  var radialmenuInstance = new ej.RadialMenu($("#defaulttradialmenu"), {
    radius: "150"
  });
});
$("#rteSampleone").select(function (e) {
  $('#defaulttradialmenu').ejRadialMenu("show");
});
```

The following screenshot illustrates the output.

Model-view-controller (MVC) is a software architecture pattern which separates the representation of information from the user's interaction with it. It consists of application data, business rules, logic, and functions. A view can be any representation of data, such as a chart or a diagram. Multiple views of the same data are used for different management and a tabular view for accountants. The controller mediates between the user and the model or view. The central ideas behind MVC are code reuse and separation of concerns. The MVC design pattern divides an application into three kinds of components, the MVC design pattern.



#### Adding events to Radial Menu items

You can specify the click event to corresponding image/text of **Radial Menu** items for performing some specific action. You need to handle the click event in script manually for each item click.

#### HTML

```
<div id="defaulttradiamenu">
<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/copy.png" data-
ej-text="Copy" data-ej-click="copy"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/Font_letter.png"
data-ej-text="Font"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/list.png" data-
ej-text="List">
<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/list.png" data-
ej-text="List"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/l5.png" data-ej-
text="List"></li>
</ul>
</li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/font.png" data-
ej-text="Bold" data-ej-click="bold">
<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/f1.png" data-ej-
text="Italic" data-ej-click="italic"></li>
```

```

<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/font.png" data-
ej-text="Bold" data-ej-click="bold"></li>
</ul>
</li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/paste.png" data-
ej-text="Paste"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/undo.png" data-
ej-text="Undo" data-ej-enabled="false">
<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/undo.png" data-
ej-text="Undo" data-ej-enabled="false"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/redo.png" data-
ej-text="Redo" data-ej-enabled="false"></li>
</ul>
</li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/align.png" data-
ej-text="Alignment">
<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/a1.png" data-ej-
text="Left"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/a2.png" data-ej-
text="Right"></li>
</ul>
</li>
</ul>
</div>

```

Add the following script in your code.

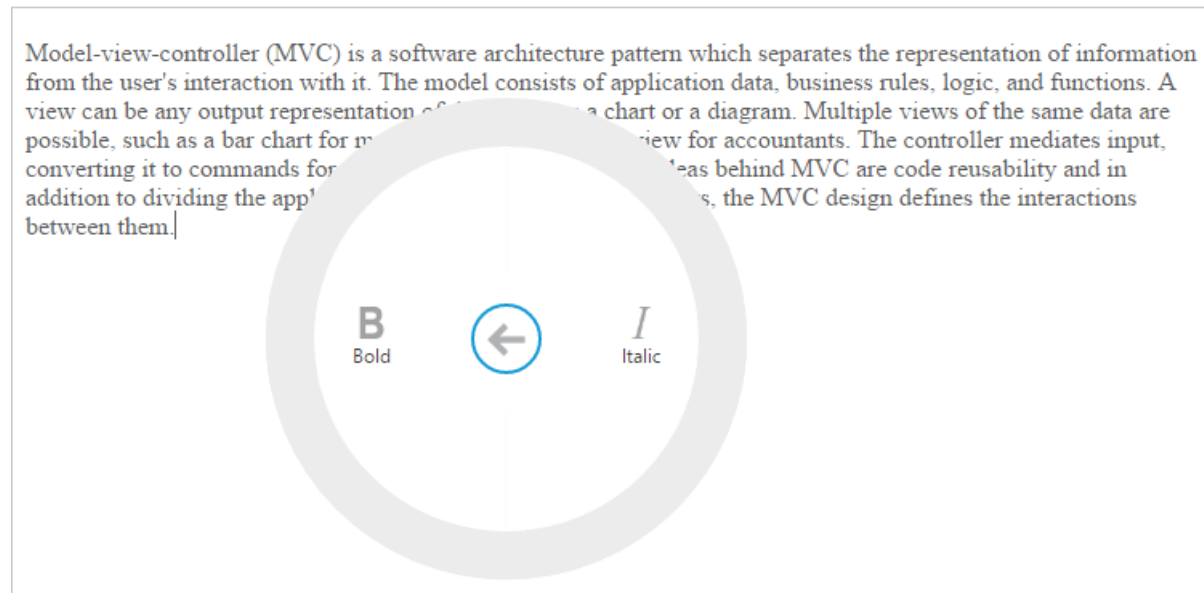
#### JAVASCRIPT

```

$(function () {
var radialmenuInstance = new ej.RadialMenu($("#defaulttradialmenu"), {
radius: "150"
});
});
$("#rteSampleone").select(function (e) {
$('#defaulttradialmenu').ejRadialMenu("show");
});
function italic(e) {
rteObj.executeCommand("italic");
}
function bold(e) {
rteObj.executeCommand("bold");
}
function copy(e) {
rteObj.executeCommand("copy");
}
}

```

The following screenshot illustrates the output.



#### Badge Customization in Radial Menu Items

You can specify set badges for the items by using badge settings in the radial menu. You can set value for the badge and enable badge with a default value.

The **data-ej-badge-enabled** property to enable or disable badges. **data-ej-badge-value** attribute is to set the badge value.

#### HTML

```
<div id="defaulttr radialmenu">
<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/copy.png" data-
ej-text="Copy" data-ej-click="copyevent" data-ej-enabled="true" data-ej-
badge-enabled="true" data-ej-badge-value="3"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/paste.png" data-
ej-text="Paste"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/redo.png" data-
ej-text="Redo"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/undo.png" data-
ej-text="Undo"></li>
</ul>
```

Add the following script in your code.

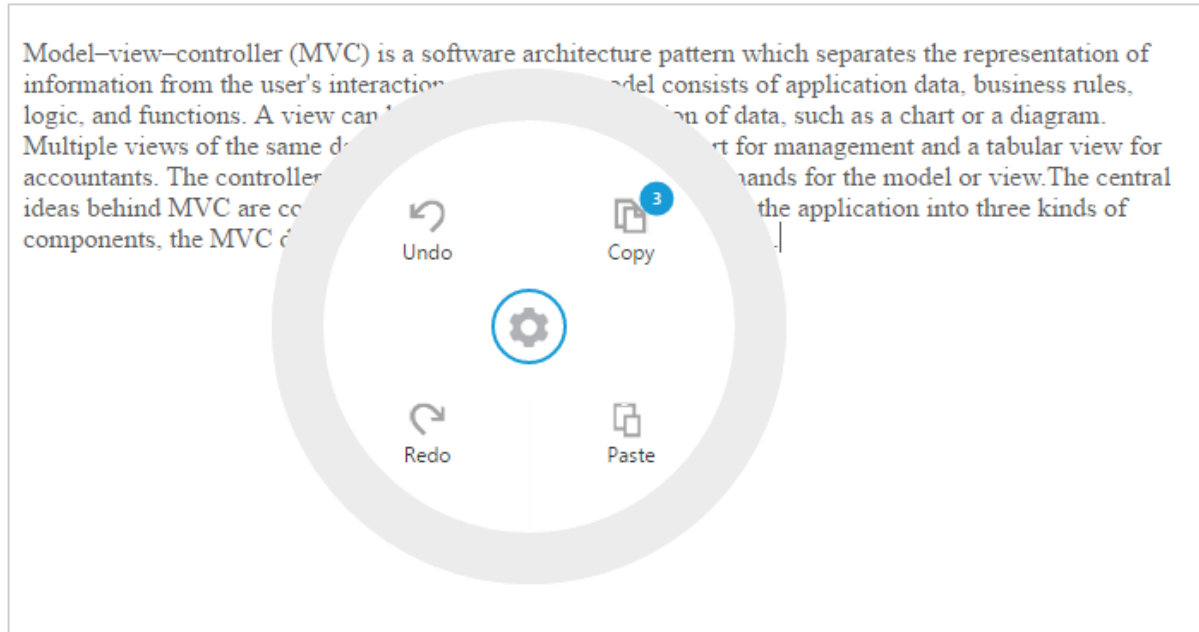
#### JAVASCRIPT

```
$(function () {
$('#defaulttr radialmenu').ejRadialMenu();
});
```



```
$("#rteSampleone").select(function (e) {
    $('#defaulttrradialmenu').ejRadialMenu("show");
});
```

The following screenshot illustrates the output.



#### Slider support for Radial Menu Items

You can customize the Radial Menu with slider settings. The **data-ej-type** property specifies the type of radial menu item, where you can set the type for RadialMenu item.

You can customize the **Radial Menu** slider settings by using the **ticks**, **strokeWidth** and **labelSpace** properties. The **data-ej-sliderSettings-ticks** property specifies the slider ticks for radial menu item. **data-ej-sliderSettings-strokeWidth** attribute is used to specify the slider's stroke width value. **data-ej-sliderSettings-labelSpace** attribute is used to specify the value of slider label space.

Refer to the following code example.

#### HTML

```
<div id="radialslidermenu">
<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/copy.png" data-
ej-text="Copy" data-ej-click="copy"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/Font_letter.png"
data-ej-text="Font" data-ej-click="font" data-ej-badge-value="2" data-ej-
badge-enabled="true" data-ej-type="slider" data-ej-slidersettings-
ticks="[0, 2, 4, 6, 8, 10, 12, 14]" data-ej-slidersettings-
strokewidth="1"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/list.png" data-
ej-text="List">
<ul>
```

```

<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/list.png" data-
ej-text="List"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/l5.png" data-ej-
text="List"></li>
</ul>
</li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/font.png" data-
ej-text="Bold">
<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/fl.png" data-ej-
text="Italic"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/font.png" data-
ej-text="Bold"></li>
</ul>
</li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/paste.png" data-
ej-text="Paste"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/undo.png" data-
ej-text="Undo" data-ej-enabled="false">
<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/undo.png" data-
ej-text="Undo" data-ej-enabled="false"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/redo.png" data-
ej-text="Redo" data-ej-enabled="false"></li>
</ul>
</li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/align.png" data-
ej-text="Alignment">
<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/a1.png" data-ej-
text="Left"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/a2.png" data-ej-
text="Right"></li>
</ul>
</li>
</ul>
</div>

```

Add the following script in your code.

#### JAVASCRIPT

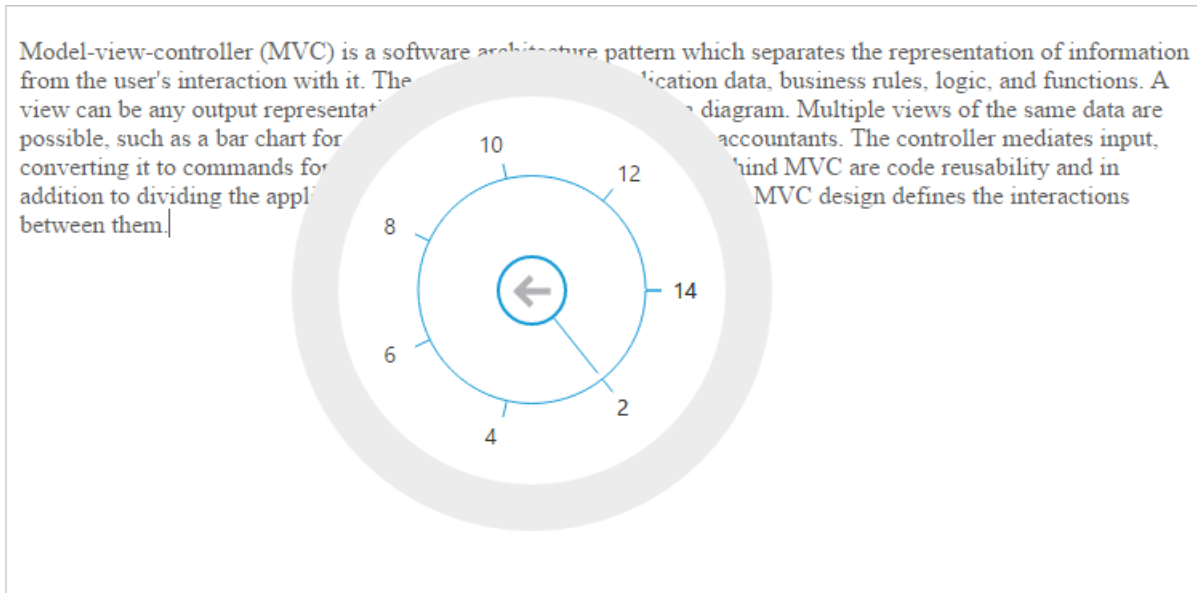
```

$(function () {
var radialmenuInstance = new ej.RadialMenu($("#radialslidermenu"), {

```

```
radius: "150"
});
});
$("#rteSampleone").select(function (e) {
$('#radialslidermenu').ejRadialMenu("show");
});
```

The following screenshot illustrates the output.



### Image Customization

You can customize the **Radial Menu's** Center and Back images by using the **ImageClass** and **BackImageClass** properties. Every menu item can be added with image using **url** property. By using this **data-ej-imageClass** attribute, you can customize the **Radial Menu** center image.

Sub-Items are also supported in the **Radial Menu**. To navigate Sub-Items, click the arrows in the outer ring and it displays the corresponding sub-items group. Clicking the center button when a sub-items group is shown, displays the items on the previous level. Nested **Radial Menu** has the second level back button. In this case, you can use the **data-ej-backImageClass** attribute to change your second level back button. **BackImageClass** is used to customize the **nestedRadialMenu** back image. Refer to the following code example.

You can add the page content with text-area by referring to this section.

### HTML

```
<div id="nestedRadialMenu">
<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/copy.png" data-
ej-text="Copy"></li>
<li data-ej-
imageurl="http://js.syncfusion.com/UG/web/Content/radial/font.png" data-
ej-text="Font">
<ul>
```

```
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/f1.png" data-ej-  
text="Italic"></li>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/f2.png" data-ej-  
text="Bold"></li>  
</ul>  
</li>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/table.png" data-  
ej-text="Table"></li>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/list.png" data-  
ej-text="List">  
<ul>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/l1.png" data-ej-  
text="List"></li>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/l2.png" data-ej-  
text="List"></li>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/l3.png" data-ej-  
text="List"></li>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/l4.png" data-ej-  
text="List"></li>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/l5.png" data-ej-  
text="List"></li>  
</ul>  
</li>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/paste.png" data-  
ej-text="Paste">  
<ul>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/c1.png" data-ej-  
text="Paste"></li>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/c2.png" data-ej-  
text="Paste"></li>  
</ul>  
</li>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/sort.png" data-  
ej-text="Sort">  
<ul>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/s1.png" data-ej-  
text="Sort"></li>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/s2.png" data-ej-  
text="Sort"></li>  
</ul>  
</li>
```

```
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/align.png" data-ej-  
ej-text="Alignment">  
<ul>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/a1.png" data-ej-  
text="Left"></li>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/a2.png" data-ej-  
text="Right"></li>  
</ul>  
</li>  
<li data-ej-  
imageurl="http://js.syncfusion.com/UG/web/Content/radial/draw.png" data-ej-  
ej-text="Draw"></li>  
</ul>  
</div>
```

Add the following script in your code.

#### **JAVASCRIPT**

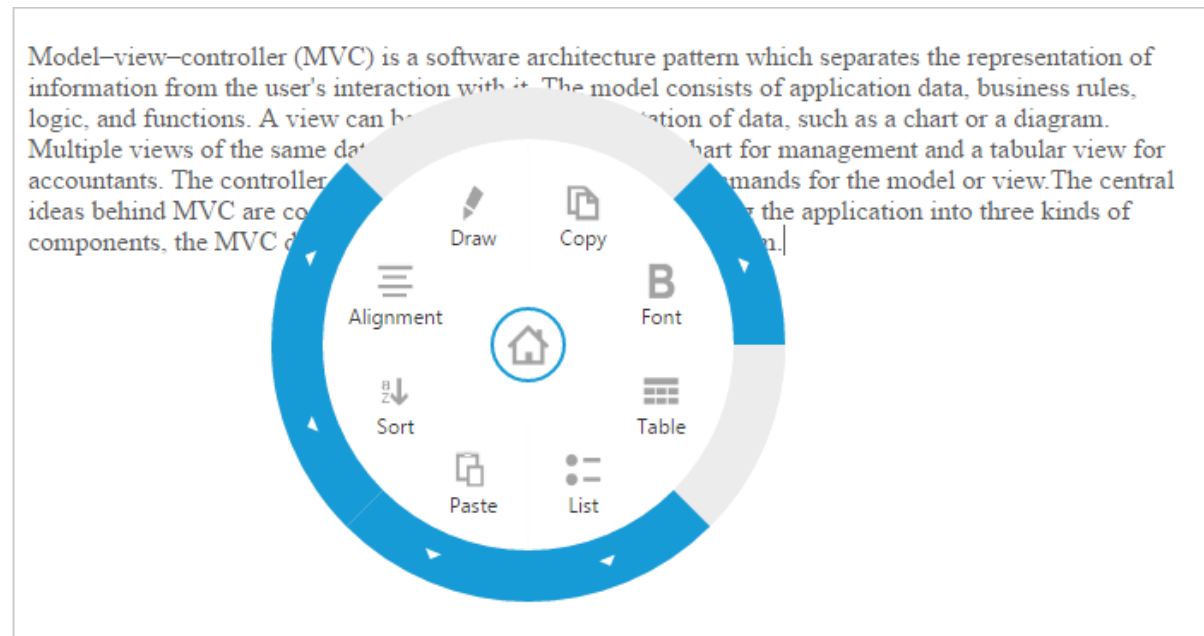
```
$(function () {  
var radialMenuInstance = new ej.RadialMenu($("#nestedRadialMenu"), {  
imageClass: "imageClass",  
backImageClass: "backImageClass" });  
});  
$("#rteSampleOne").select(function (e) {  
$('#nestedRadialMenu').ejRadialMenu("show");  
});
```

Add the following styles in your code.

#### **CSS**

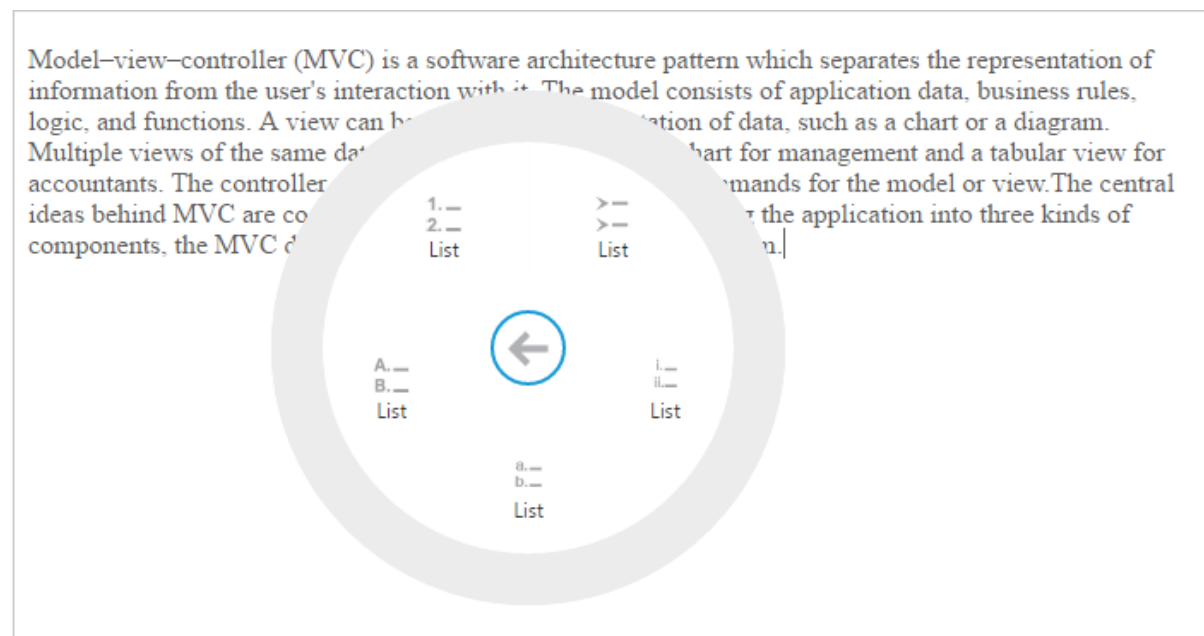
```
<style type="text/css" class="cssStyles">  
.e-radialmenu .imageClass {  
background-image:  
url(http://js.syncfusion.com/UG/web/Content/radial/main.png);  
}  
.e-radialmenu .backImageClass {  
background-image:  
url(http://js.syncfusion.com/UG/web/Content/radial/Back_button.png);  
}  
</style>
```

The following screenshot illustrates the output.



Radial Menu - Image Customization – Main menu

When you click the arrow, it navigates to the child item as illustrated in the following screenshot.



Radial Menu- Image Customization – Child menu

### Appearance and Styling

You can customize RadialMenu control style and the appearance by using available themes or cssClass property.

#### Theme

In order to apply styles to the RadialMenu control, refer these 2 files namely, ej.widgets.core.min.css and ej.theme.min.css. When you refer ej.web.all.min.css file, it is not necessary to include the files

ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.web.all.min.css is the combination of these two.

By default, there are 16 theme support available for RadialMenu component, namely:

- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap
- high-contrast-01
- high-contrast-02
- material
- office-365

### CSS Class

RadialMenu component's appearance can only be customized using its CSS classes. Define CSS class, as per requirement and assign the class name to cssClass property.

#### *Configure RadialMenu using CSS class*

Define CSS class to customize the RadialMenu control.

### CSS

```
<style type="text/css">
/* Customize the radialmenu */
.e-radialmenu .e-default, .e-radialmenu .e-outerdefault.customCss
{
fill:#f00;
}
</style>
```

In the HTML page, define the li items for RadialMenu component.

### HTML

```
<div id="radialmenu">
<ul>
<li data-ej-
imageurl="http://js.syncfusion.com/ug/web/content/radial/copy.png"
data-ej-text="Copy">
</li>
<li data-ej-
imageurl="http://js.syncfusion.com/ug/web/content/radial/paste.png"
data-ej-text="Paste">
```

```

</li>
<li data-ej-
imageurl="http://js.syncfusion.com/ug/web/content/radial/redo.png"
data-ej-text="Redo">
</li>
<li data-ej-
imageurl="http://js.syncfusion.com/ug/web/content/radial/undo.png"
data-ej-text="Undo">
</li>
</ul>
</div>

```

Initialize the **Radial Menu** control with `cssClass` in the script as follows.

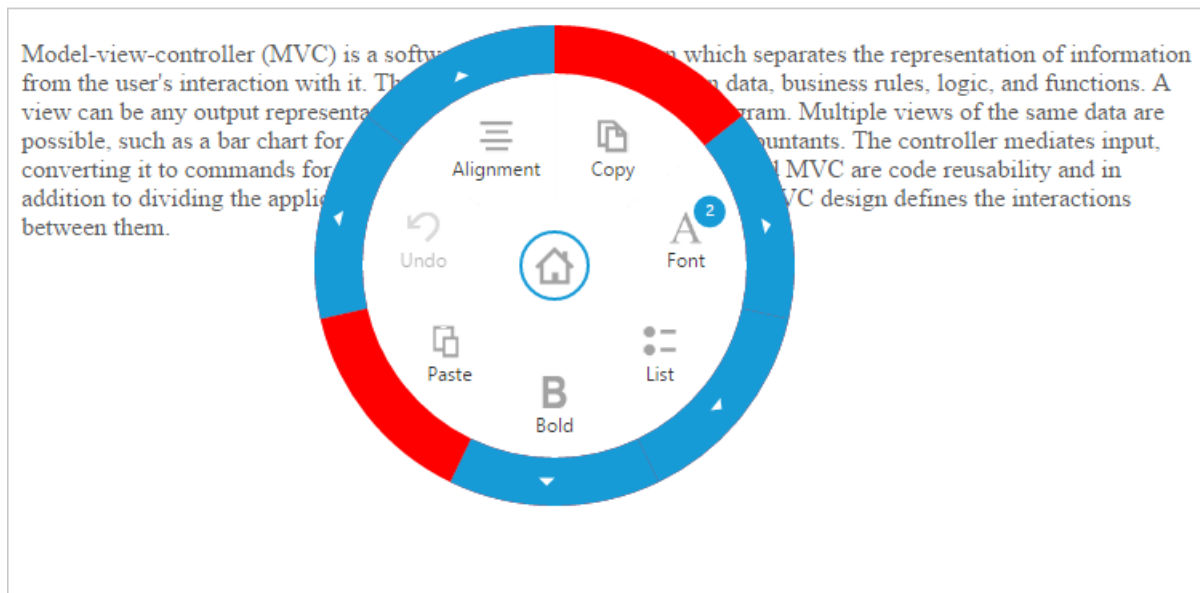
### JAVASCRIPT

```

module RadialMenuComponent {
$(function () {
var radialmenuInstance = new ej.RadialMenu($("#radialmenu"), {
cssClass: "customCss"
});
});
}

```

Output of RadialMenu configured based on CSS class is as follow,



### Template Support

Template support for RadialMenu items will allow you to use any type of `<svg>` permitted tags inside our template. Here for example, using this template support you can use the SVG icons in Radial Menu instead of image tags. To use SVG icons in RadialMenu, you need to use `prependTo` property.

#### Add SVG to item Icon

Using SVG icon will optimize the icons quality and to reduce space occupation by normal images and svg images are scalable and zoom. Define the text element for SVG with x and y position and code for rendering the font icons. Assign the SVG element ID to **prependTo** property.



**HTML**

```

<div id="radialtarget1" class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<textarea id="rteSample1" rows="10" cols="70" style="height: 440px">
<p>
Model-view-controller (MVC) is a software architecture pattern which
separates the representation of information from the user's interaction
with it.
The model consists of application data, business rules, logic, and
functions. A view can be any output representation of data, such as a
chart or a diagram.
Multiple views of the same data are possible, such as a bar chart for
management and a tabular view for accountants.
The controller mediates input, converting it to commands for the model or
view.The central ideas behind MVC are code reusable and in addition to
dividing the application into three kinds of components, the MVC design
defines the interactions between them.
</p>
<p>A controller can send commands to its associated view to change the
view's presentation of the model (e.g., by scrolling through a document).
It can also send commands to the model to update the model's state (e.g.,
editing a document).</p>
<p>A model notifies its associated views and controllers when there has
been a change in its state. This notification allows the views to produce
updated output, and the controllers to change the available set of
commands. A passive implementation of MVC omits these notifications,
because the application does not require them or the software platform
does not support them.</p>
<p>A view requests from the model the information that it needs to
generate an output representation to the user.</p>
</textarea>
</div>
</div>
</div>
<div id="defaultRadialMenu">
<ul>
<li data-ej-prependTo="#template1" data-ej-text="Bold" data-ej-
click="bold"></li>
<li data-ej-prependTo="#template2" data-ej-text="Italic" data-ej-
click="italic"></li>
<li data-ej-prependTo="#template3" data-ej-text="Redo" data-ej-
click="redo" data-ej-enabled="false"></li>
<li data-ej-prependTo="#template4" data-ej-text="Undo" data-ej-
click="undo" data-ej-enabled="false"></li>
</ul>
</div>
<svg id="template1" style="display: none"
xmlns="http://www.w3.org/2000/svg">
<text x="210" y="100">&#xe636;</text>
</svg>
<svg id="template2" style="display: none"
xmlns="http://www.w3.org/2000/svg">
<text x="210" y="218">&#xe635;</text>
</svg>

```

```
<svg id="template3" style="display: none"
xmlns="http://www.w3.org/2000/svg">
<text x="90" y="215">&#xe606;</text>
</svg>
<svg id="template4" style="display: none"
xmlns="http://www.w3.org/2000/svg">
<text x="93" y="100">&#xe607;</text>
</svg>
```

Now add the following in your Script section,

#### JAVASCRIPT

```
<script type="text/javascript">
var rteObj, rteElement = $("#rteSample1"), radialElement =
$('#defaultRadialMenu'), action = 0, forRedo = 0;
$(function () {
rteElement.ejRTE({ width: "100%", minWidth: "10px", change: "rteChange",
select: "radialShow", showToolBar: false, showContextMenu: false });
rteObj = rteElement.data("ejRTE");
if (!(ej.browserInfo().name == 'msie' && ej.browserInfo().version < 9)) {
radialElement.ejRadialMenu({ imageClass: "imageClass", backImageClass:
"backImageClass", targetElementId: "radialtarget1" });
$("#radialtarget1").parent().css("position", "relative");
}
else {
$("#contentDiv").html("Radial Menu is only supported from Internet
Explorer Versioned 9 and above.").css({ "font-size": "20px", "color":
"red" });
}
$(window).resize(function () {
if(ej.isMobile() && ej.isPortrait())
$('#defaultRadialMenu').css({ "left": 25 });
});
function radialShow(e) {
var target = $("#radialtarget1"), radialRadius = 150, radialDiameter = 2
* radialRadius,
// To get Iframe positions
iframeY = e.event.clientY, iframeX = e.event.clientX,
// To set Radial Menu position within target
x = iframeX > target.width() - radialRadius ? target.width() -
radialDiameter : (iframeX > radialRadius ? iframeX - radialRadius : 0),
y = iframeY > target.height() - radialRadius ? target.height() -
radialDiameter : (iframeY > radialRadius ? iframeY - radialRadius : 0);
radialElement.ejRadialMenu("setPosition", x, y);
radialElement.focus();
$('iframe').contents().find('body').blur();
}
function rteChange(e) {
radialElement.ejRadialMenu("enableItem", "Undo");
}
function bold(e) {
rteObj.executeCommand("bold");
data = rteObj._getSelectedHtmlString() ? true : false;
if (data) action += 1;
}
```

```

forRedo = action;
radialElement.focus();
}
function italic(e) {
rteObj.executeCommand("italic");
data = rteObj._getSelectedHtmlString() ? true : false;
if (data) action += 1;
forRedo = action;
radialElement.focus();
}
function undo(e) {
rteObj.executeCommand("undo");
action -= 1;
if (action == 0)
radialElement.ejRadialMenu("disableItem", "Undo");
radialElement.ejRadialMenu("enableItem", "Redo");
radialElement.focus();
}
function redo(e) {
rteObj.executeCommand("redo");
action += 1;
if (forRedo == action) radialElement.ejRadialMenu("disableItem", "Redo");
radialElement.ejRadialMenu("enableItem", "Undo");
radialElement.focus();
}
</script>

```

Add the following in style section,

### CSS

```

<style>
.e-radialmenu .imageClass {
background-image: url(../content/images/RadialMenu/settings.png);
}
@font-face {
font-family: 'ej-webfont';
src: url('../content/ejthemes/common-images/icons.eot');
src: url('../content/ejthemes/common-images/icons.ttf')
format('truetype'),
url('../content/ejthemes/common-images/icons.woff') format('woff'),
url('../content/ejthemes/common-images/icons.svg') format('svg');
font-weight: normal;
font-style: normal;
font-size: 20px;
}
.e-radialmenu .e-abs.e-radialshow, .e-radialmenu .e-abs.e-scaleshadow {
/* use !important to prevent issues with browser extensions that change fonts */
font-family: 'ej-webfont' !important;
speak: none;
font-size: 20px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;

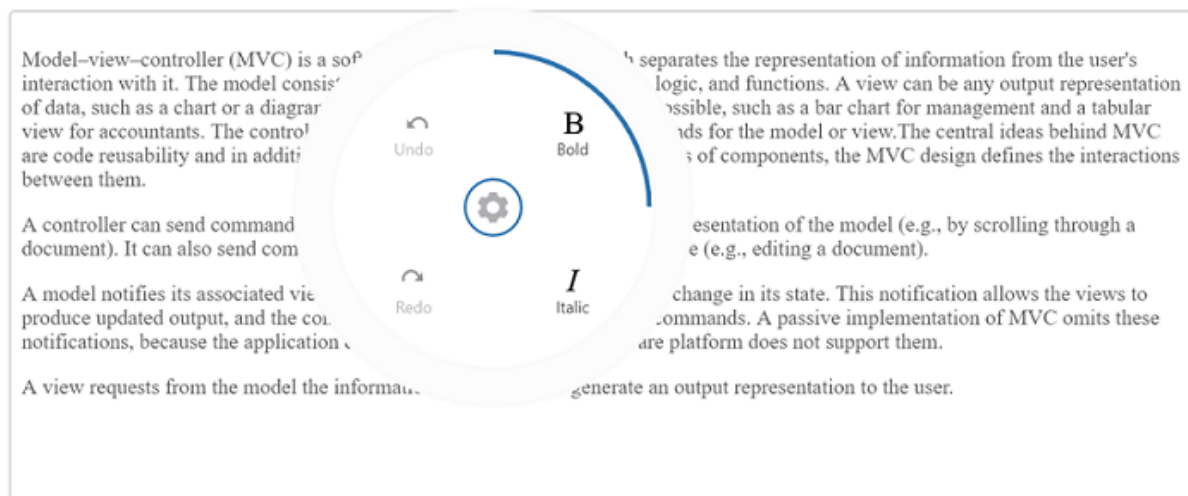
```

```

line-height: 1;
-webkit-font-smoothing: antialiased;
/* Better Font Rendering ===== */
-webkit-font-smoothing: antialiased;
}
.e-bold:before {
content: "\e636";
}
.e-italic:before {
content: "\e635";
}
.e-icon.e-redo:before {
content: "\e606";
}
.e-icon.e-undo:before {
content: "\e607";
}
}
</style>

```

The following screenshot illustrates the output,



**Note:** This is the example sample for SVG icon support for Radial Menu. Like wise you can add any SVG element to it, but you need to customize and position the element individually.

## RadialSlider

### Overview

The Typescript Radial Slider control provides an optimized interface for selecting a numeric value using touch interface. This allows the user to select a value or range of values in a circular motion.

### Key Features

**Angle Support:** Provides start and end angle level view of Radial Slider. **Animation:** Offers animation effect for the Radial Slider handle. **Image Customization:** Supports customize the images of the inner circle in Radial Slider. **Dimension:** Allows to change the radius and Stroke width of Radial Slider. \*

**Accuracy:** Provides way to select accurate numeric value and customize the display values.

## Getting Started

This section helps you to understand the getting started of the Radial Slider control with the step-by-step instructions.

### Create a Radial Slider control

The following steps guide you to add a Radial Slider control.

- 1) Refer the common Typescript [Getting Started Documentation](#) to create an application and add necessary scripts and styles for rendering the RadialSlider control. 2) Create an HTML page and add the scripts and css references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Typescript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/fl
at-azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.
web.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add Radial Slider code here-->
</body>
</html>
```

- 3) Add the following code with in the tag to render the RadialSlider control.

#### HTML

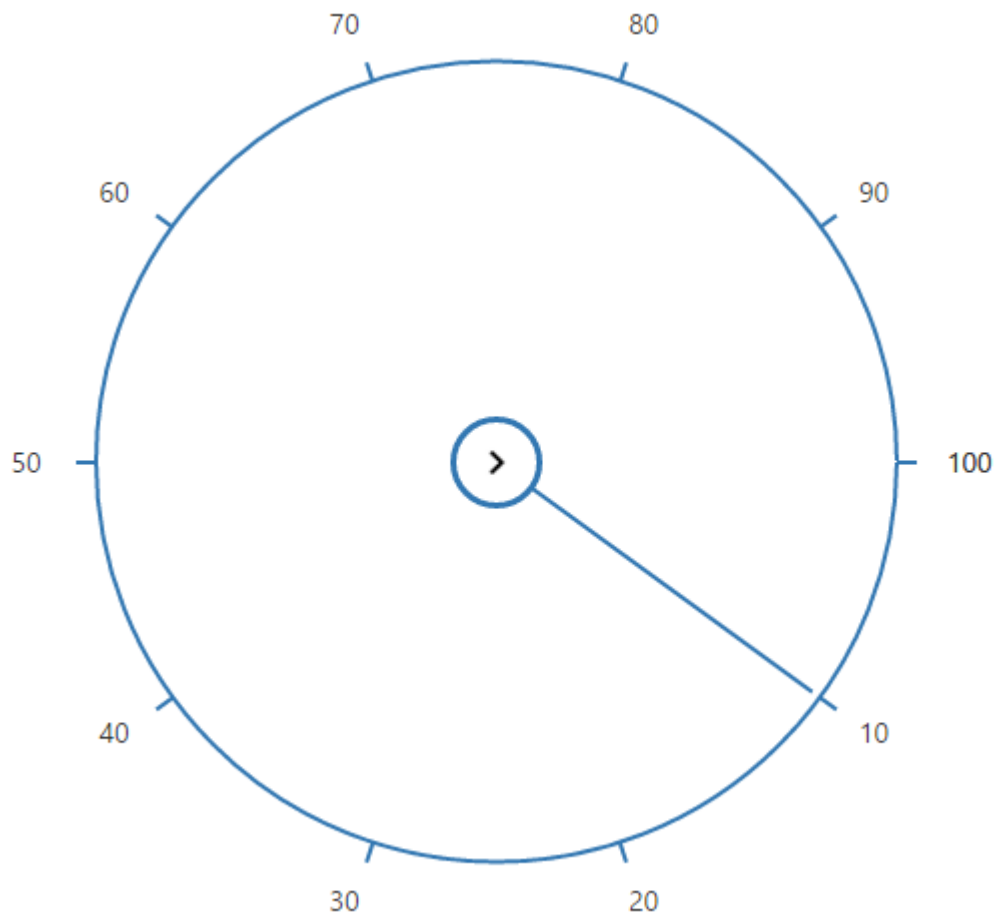
```
<div id="radialSlider">
</div>
```

Initialize the Radial Slider control in ts file by using the ej.RadialSlider method.

#### TS

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module RadialSliderComponent {
$(function () {
var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {
innerCircleImageUrl: "
http://js.syncfusion.com/demos/web/content/images/radialslider/chevron-
right.png"
});
});
}
```

To get the following output from the above-mentioned code.



---

**Note:** You can find the Radial Slider control properties from the [API reference](#).

---

### Dimension

#### Stroke Width

**Radial Slider** `strokeWidth` property specifies the width of the outline . By default, the **Radial slider** `strokeWidth` is set as 2. Refer to the following code example.

#### HTML

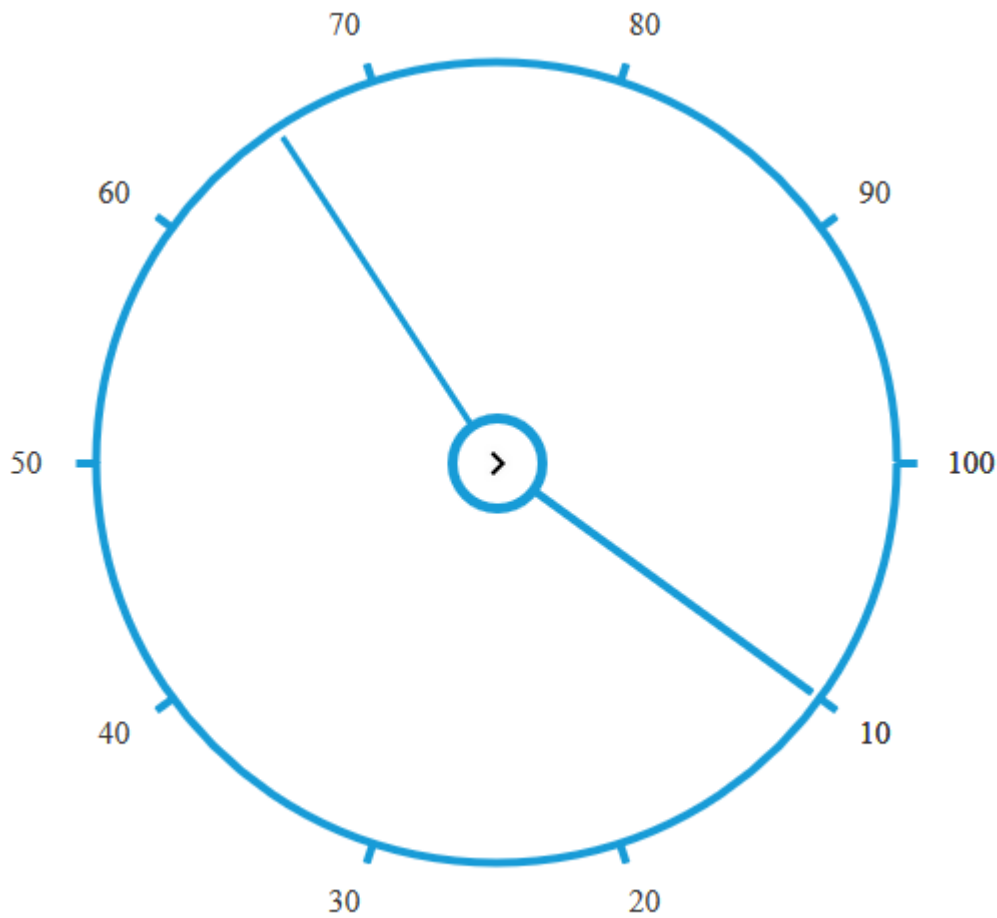
```
<div id="radialSlider">  
</div>
```

Add the following script in your code.

#### JS

```
$(function () {  
  var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {  
    innerCircleImageUrl:"chevron-right.png",  
    strokeWidth:4  
  });  
});
```

The following screenshot illustrates the output of the above code.



#### Setting radius

The **RadialSlider** property **radius** indicates the radius of the RadialSlider's circle and its allow to change radius value. By default, the **Radial Slider** radius is set to 200. Refer to the following code example.

#### HTML

```
<div id="radialSlider"> </div>
```

#### JS

```
$(function () {  
  var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {  
    innerCircleImageUrl: "chevron-right.png",  
    radius:250  
  });  
});
```

## Behavior settings

The following are some properties that enable you to have an option to enhance the behavior of **RadialSlider** control.

### Show/hide RadialSlider on initial rendering

The **RadialSlider** property **autoOpen** allows you to make the control visible on initial rendering. By default, the value of the property is set as **true**. Setting it as **false** will hide the control on initial rendering.

#### HTML

```
<div id="radialSlider"> </div>
```

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RadialSliderComponent {
    $(function () {
        var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {
            innerCircleImageUrl: "chevron-right.png",
            autoOpen: false
        });
    });
}
```

### Value accuracy

The **RadialSlider** property **enableRoundOff** allows to show the rounded off value when user changes it. By default, the value of the property is set as **true**. For accurate values, you can set this as **false**.

#### HTML

```
<div id="radialSlider"> </div>
```

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RadialSliderComponent {
    $(function () {
        var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {
            innerCircleImageUrl: "chevron-right.png",
            enableRoundOff: false
        });
    });
}
```

### Display inline

The **RadialSlider** property **inline** is used to show the control values inline of the slider. By default, the value of the property is set as **false**.

#### HTML

```
<div id="radialSlider"> </div>
```



### JS

```
$(function () {  
  var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {  
    innerCircleImageUrl: "chevron-right.png",  
    inline:true  
  });  
});
```

### Modifying Label Space

The **RadialSlider** property **labelSpace** allow to define the space of label in it. By default, the **RadialSlider** **labelSpace** is set as 30. Refer to the following code example.

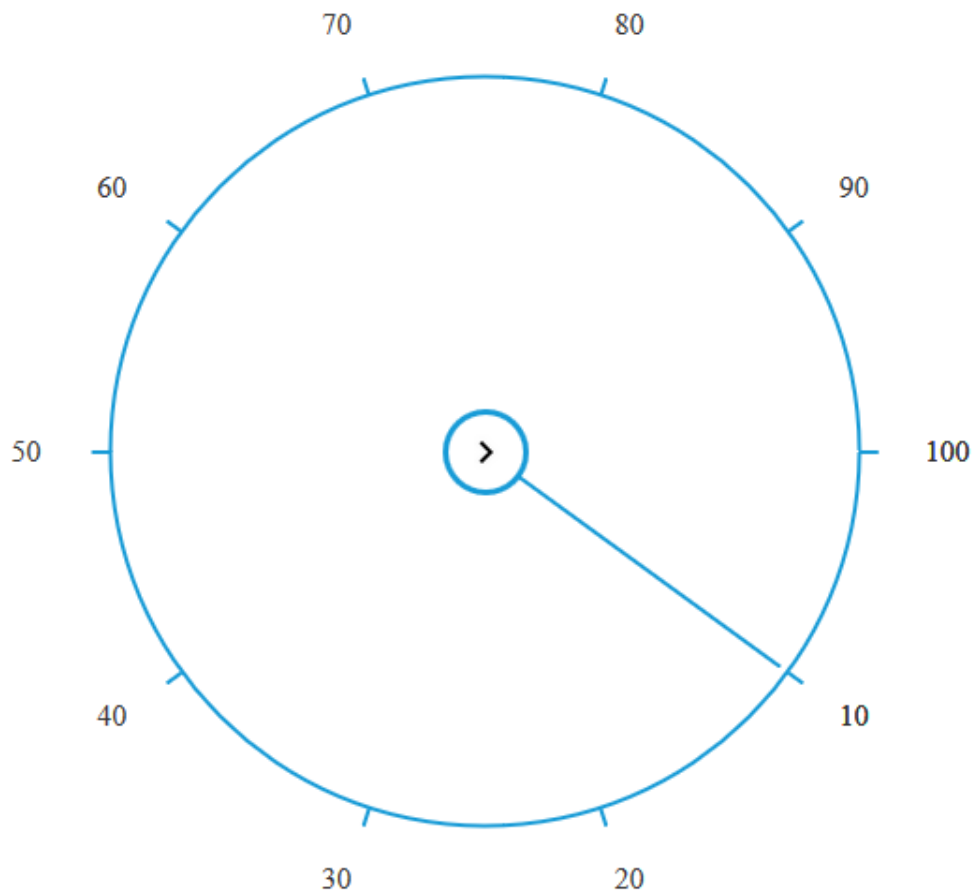
### HTML

```
<div id="radialSlider"> </div>
```

### JS

```
$(function () {  
  var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {  
    innerCircleImageUrl: "chevron-right.png",  
    labelSpace:40  
  });  
});
```

The following screenshot shows the output for the above code example.



#### Show/Hide inner circle

The **RadialSlider** property **showInnerCircle** allow to show or hide the inner circle of the **RadialSlider**. By default, Value of the property is set as **true** and type of the property is **Boolean**.

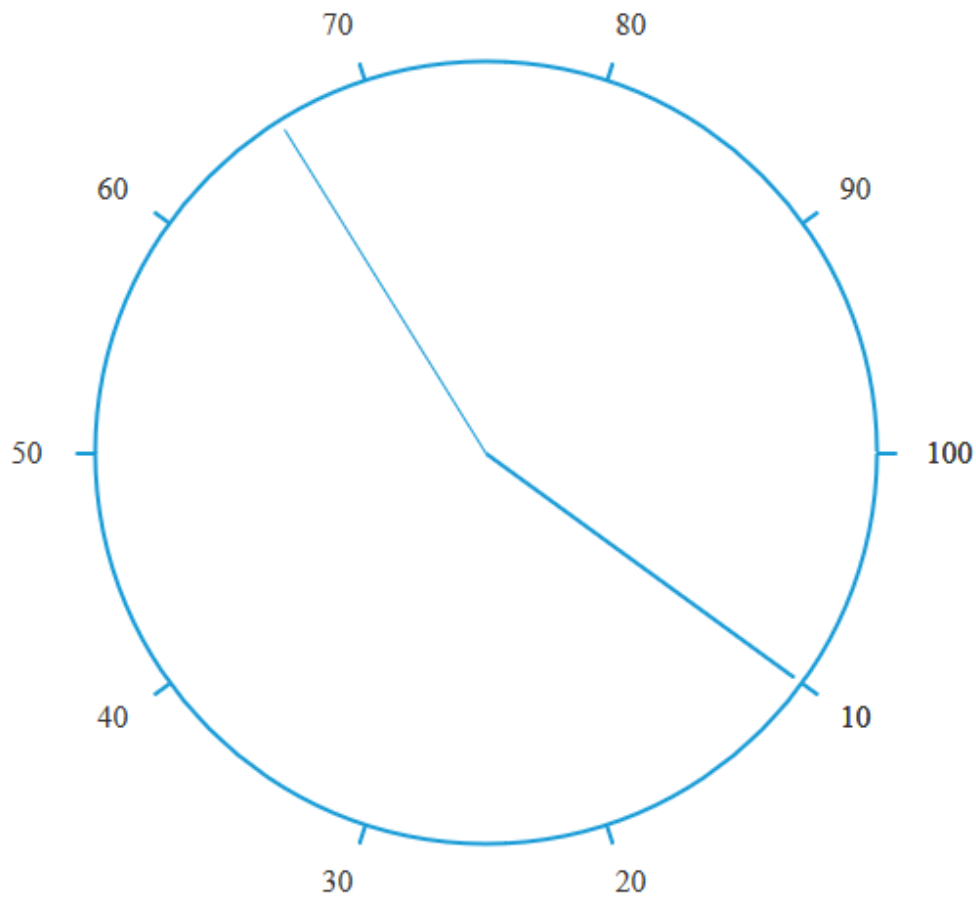
#### HTML

```
<div id="radialSlider"> </div>
```

#### JS

```
$(function () {  
  var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {  
    innerCircleImageUrl: "chevron-right.png",  
    showInnerCircle: false  
  });  
});
```

The following screenshot shows the output for the above code example.



### Values customization

The **RadialSlider** property **ticks** allow to set an array of a numeric value which will be displayed in it. By Default **RadialSlider** ticks value is a set to an array of length 11 as following,

ticks = { 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 }. You can refer the below code for reference.

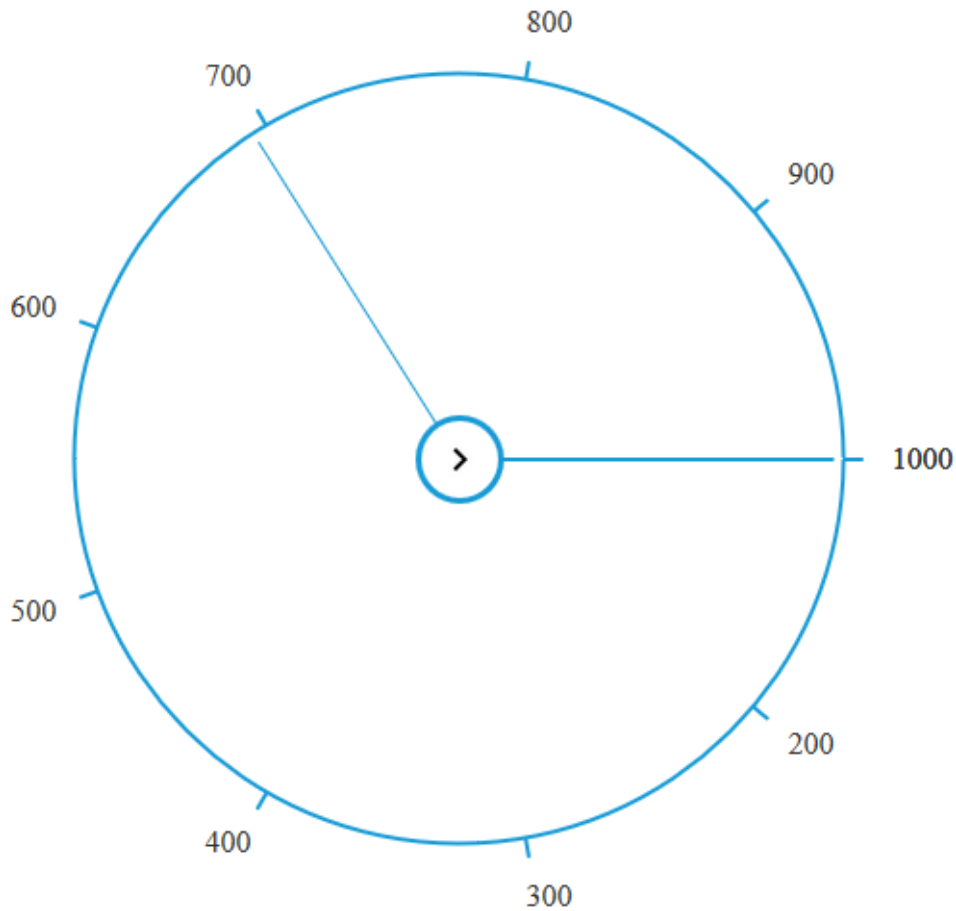
### HTML

```
<div id="radialSlider"> </div>
```

### JS

```
$(function () {  
  var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {  
    innerCircleImageUrl: "chevron-right.png",  
    ticks:[100,200,300,400,500,600,700,800,900,1000],  
    value:100  
  });  
});
```

The following screenshot shows the output for the above code example.



## Image customization

### Customizing inner circle

#### Using Class

The **RadialSlider** property **innerCircleImageClass** allow to set image for the inner circle of the **RadialSlider**. By default, the **Radial Slider** innerCircleImageClass is set as "null". Refer to the following code example.

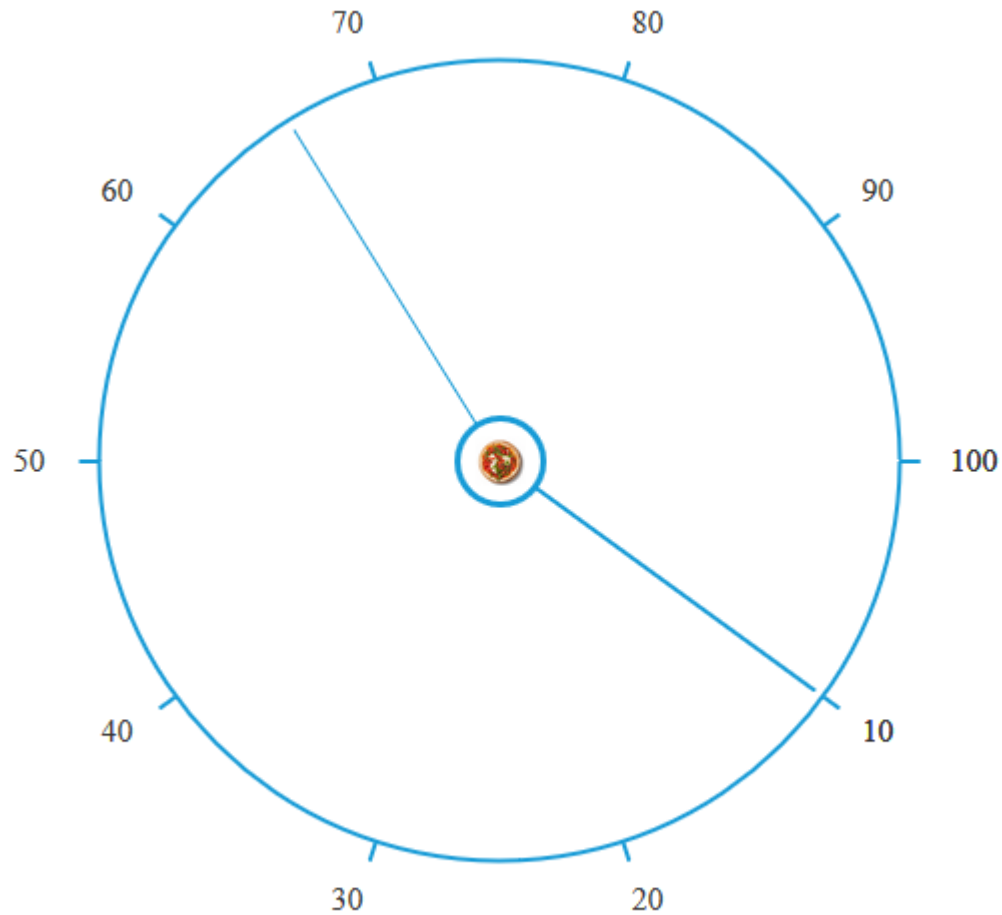
#### HTML

```
<div id="radialSlider"> </div>
```

#### JS

```
$(function () {  
  var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {  
    innerCircleImageClass:"Radial-Slider"  
  });  
});
```

The following screenshot illustrates the output of above code.



#### Using image URL

The **RadialSlider** property **innerCircleImageUrl** allow to set URL image to the inner circle of the **RadialSlider**. By default, the **Radial Slider** innerCircleImageUrl is set as "null". Refer to the following code example.

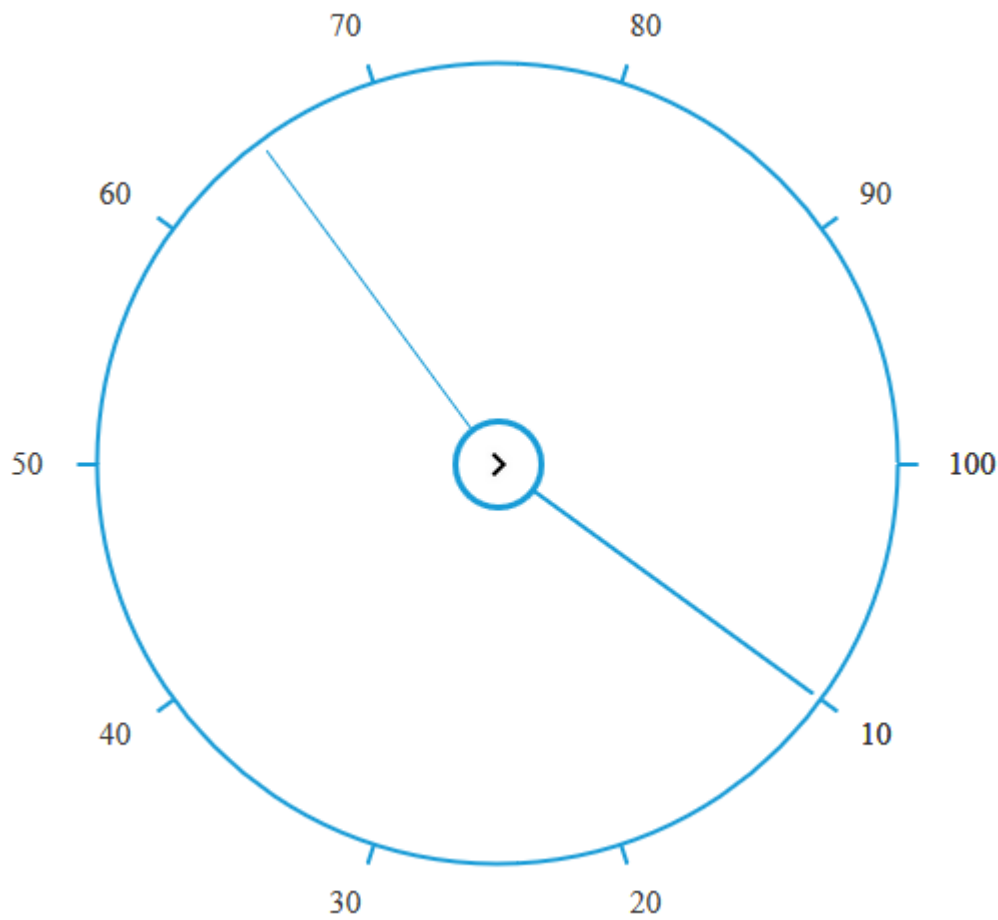
#### HTML

```
<div id="radialSlider"> </div>
```

#### JS

```
$(function () {  
  var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {  
    innerCircleImageUrl:"chevron-right.png"  
  });  
});
```

The following screenshot illustrates the output of above code.



### Animation Effect

The animation effect for the **RadialSlider** can be enabled or disabled using the **enableAnimation** property. By default, it is set as **true**, so that the handle movement will be animated.

### HTML

```
<div id="radialSlider"></div>
```

Add the following script in your code and this will stop the handle movement's animation.

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RadialSliderComponent {
  $(function () {
    var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {
      innerCircleImageUrl: "chevron-right.png",
      enableAnimation: false
    });
  });
}
```

## Display Angle Support

### Start Angle

The **RadialSlider** property **startAngle** allows you to change the startAngle level of the **RadialSlider**. By default, the **Radial Slider** startAngle is set as 0. Refer to the following code example.

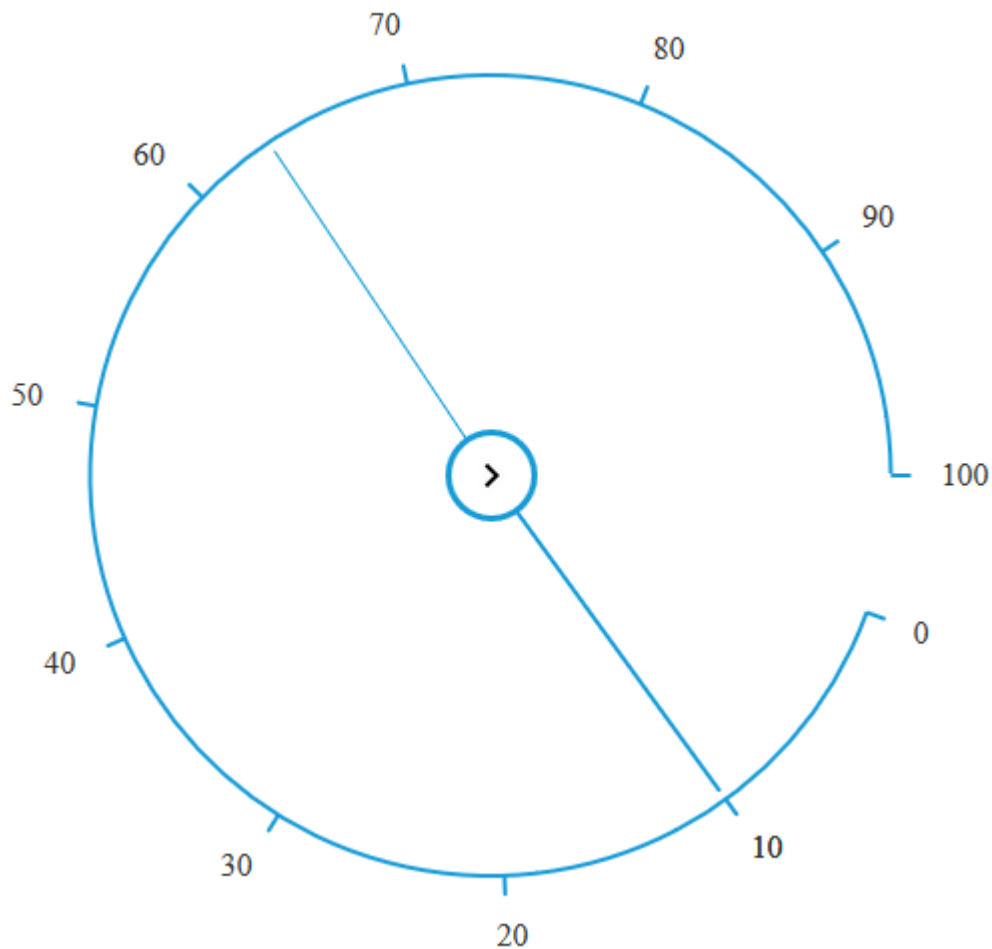
#### HTML

```
<div id="radialSlider">  
</div>
```

#### JS

```
$(function () {  
  var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {  
    innerCircleImageUrl:"chevron-right.png",  
    startAngle: 20  
  });  
});
```

The following screenshot illustrates the output of the above code.



### End Angle

The **RadialSlider** property **endAngle** allows you to change the endAngle level of the **RadialSlider**. By default, the **Radial Slider** endAngle is set as 360. Refer to the following code example.

### HTML

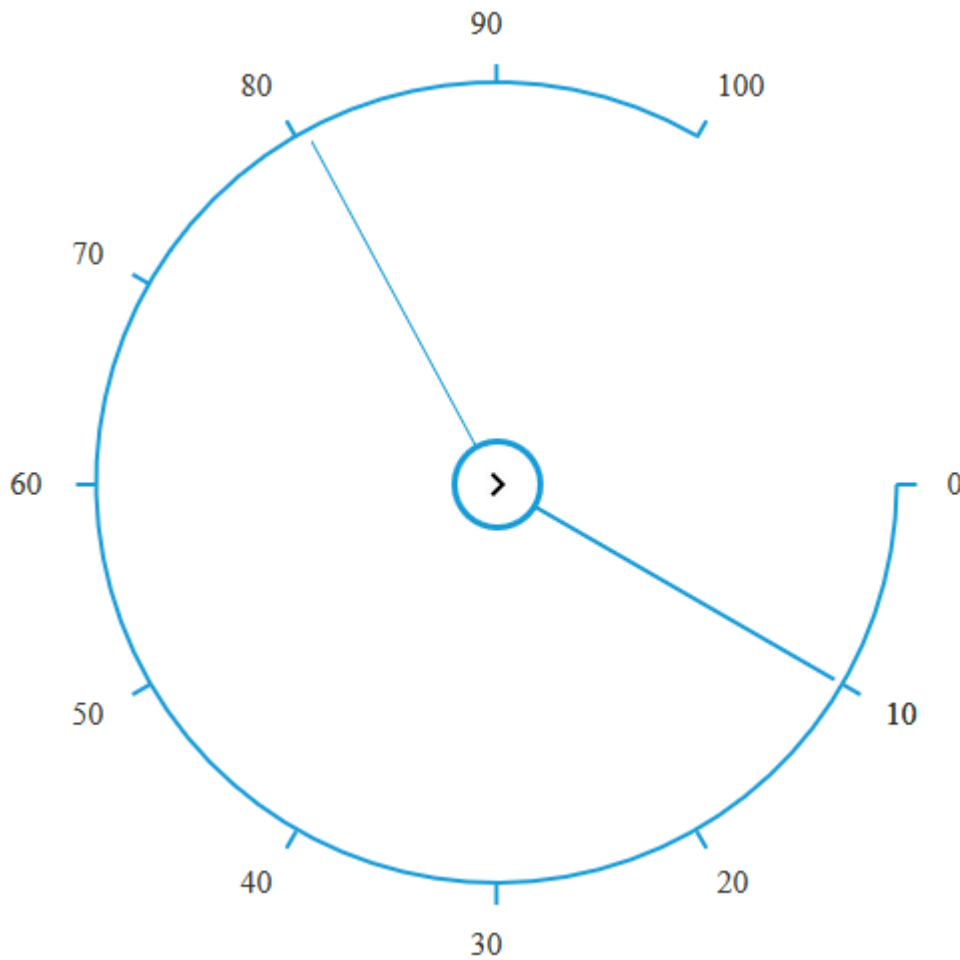
```
<div id="radialSlider"></div>
```

### JS

```
$(function () {  
  var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {  
    innerCircleImageUrl: "chevron-right.png",  
    endAngle: 300  
  });  
});
```

The following screenshot illustrates the output of the above code.





## Appearance and Styling

### Theme

You can customize RadialSlider control style and the appearance by using available themes or `cssClass` property.

### Theme

In order to apply styles to the RadialSlider control, refer to 2 files namely, `ej.widgets.core.min.css` and `ej.theme.min.css`. When you refer `ej.web.all.min.css` file, it is not necessary to include the files `ej.widgets.core.min.css` and `ej.theme.min.css` in your project, as `ej.web.all.min.css` is the combination of these two.

By default, there are 16 theme support available for RadialSlider component, namely:

- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark

- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap
- high-contrast-01
- high-contrast-02
- material
- office-365

### CSS Class

**RadialSlider** control also allows you to customize its appearance using user-defined CSS. To apply custom themes you can use **cssClass** property, which sets the root class for **RadialSlider** theme.

Using this **cssClass** you can override the existing styles under the theme style sheet. In the following example, the value of **cssClass** property is set as **"Purple-dark"**. **Purple-dark** is added as root class to **RadialSlider** control at the runtime. From this root class, you can customize the **RadialSlider** appearance.

Add the following code in your **HTML** page to render the RadialSlider.

### HTML

```
<div id="radialSlider">
</div>
<script>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RadialSliderComponent {
$(function () {
var radialsliderInstance = new ej.RadialSlider($("#radialSlider"), {
innerCircleImageUrl:"chevron-right.png",
cssClass:"Purple-dark"
});
});
}
</script>
```

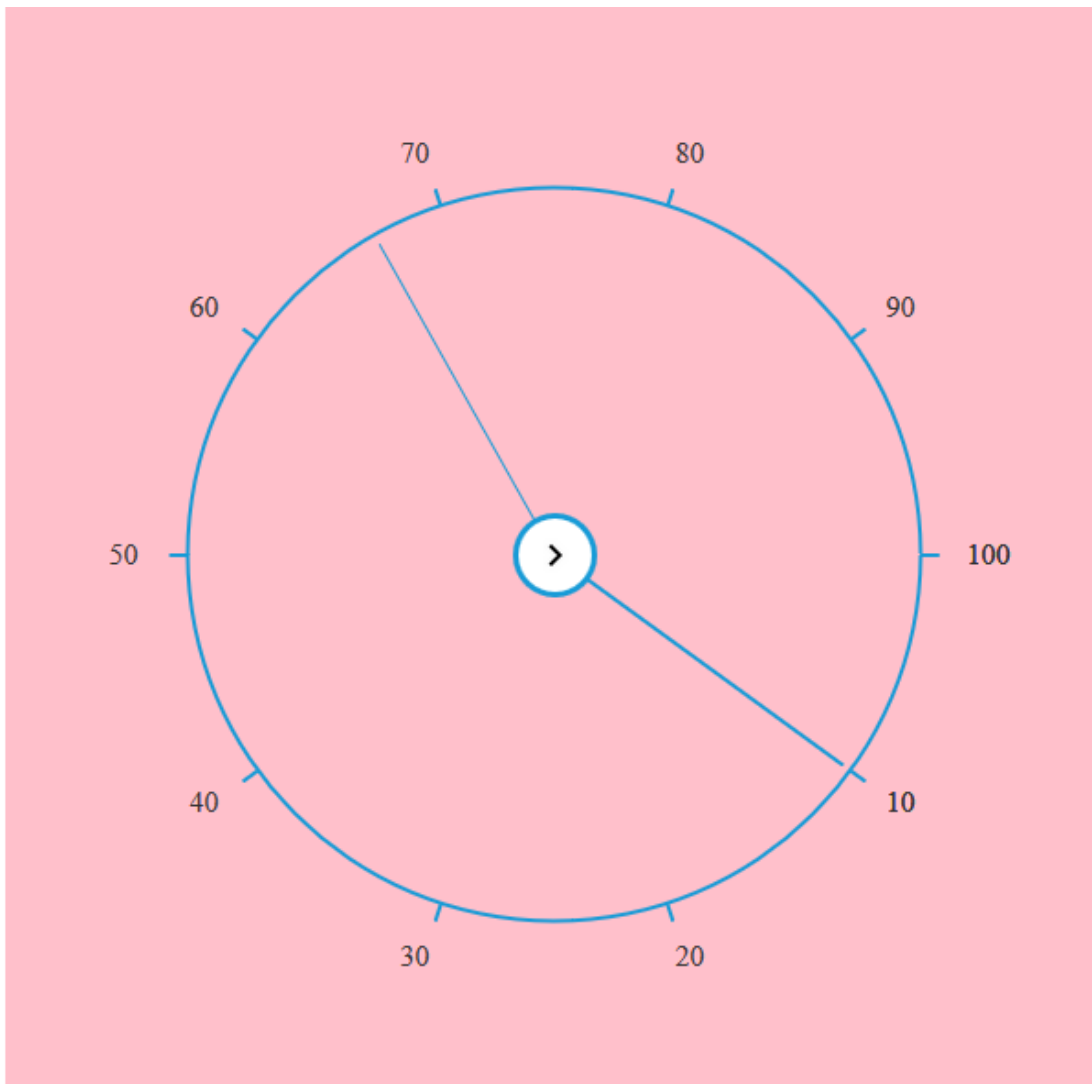
In the following style sheet the exiting theme style sheet file has been over-ridden using root class **"Purple-dark"**.

Add the following code in your style section.

### CSS

```
.Purple-dark{
background-color:pink;
}
```

The following screenshot illustrates the output of the above code.



## RadioButton

### Overview

The **RadioButton** component allows you to pick an option from multiple options to perform an action based on selection.

### Key Features

- **Trendy Look** : Rich Appearance with Theme Support
- **RTL** : Supports for right to left alignment
- **Easy Customization**: The customization of **RadioButton** is made simple
- **Themes**: Supports 17 built-in themes (6 – flat and 6 – gradient effects), and also support custom skin options to set user-defined themes.
- **Responsive and Touch support**: Fits in all range of devices and supports touch devices.

## Getting Started

This section discloses the details on how to render and configure a RadioButton component in a TypeScript application.

### Create your first Radio Button

1. Create a TypeScript application and refer the dependent modules, script and CSS with the help of given Getting started document.
2. In the index.HTML file, add the input element for rendering RadioButton component as given below.

### HTML

```
<div>
<br />
Category
<br />
<br />
<table >
<tr>
<td>
<input type="radio" id="Radio1" />
<label for="Radio1">Fresher</label>
</td>
<td colspan="2">
<input type="radio" id="Radio2" />
<label for="Radio2">Experienced</label>
</td>
</tr>
</table>
<br />
<br />
Experienced
<br />
<br />
<table>
<tr>
<td>
<input type="radio" id="Radio3" />
<label for="Radio3">1+ years</label>
</td>
<td colspan="2">
<input type="radio" id="Radio4" />
<label for="Radio4">2.5+years</label>
</td>
<td colspan="2">
<input type="radio" id="Radio5" />
<label for="Radio5">5+years</label>
</td>
</tr>
</table>
</div>
```

3. Create a TypeScript file named "app.ts" file and refer the required definition files as given below.

**JS**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

4. Now, initialize the RadioButton component by using ej.RadioButton method.

**JS**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module AppComponent {
  $(function () {
    new ej.RadioButton($("#Radio1"), {value:"Fresher", name:"Category",
    size:"small" });
    new ej.RadioButton($("#Radio2"), {value:"Experienced", name:"Category",
    size: "small", checked:true });
    new ej.RadioButton($("#Radio3"), {value:"1+ years", name:"Experienced",
    size:"medium", checked: true });
    new ej.RadioButton($("#Radio4"), {value:"2.5+ years",
    name:"Experienced",size:"medium" });
    new ej.RadioButton($("#Radio5"), {value:"5+ years", name:"Experienced",
    size:"medium"});
  }
}
```

*Run the application*

To run the application, navigate the project folder and open command prompt window and execute the following command.

**JS**

```
tsc
```

This command compiles the app.ts file to generate a JS file named app.js file.

Refer the app.js file in index.html and browse the HTML file to see the following output.

Category

☐ Fresher ☒ Experienced

Experienced

☒ 1+ years ☐ 2.5+years ☐ 5+years

## Easy Customization

### Checked status

You have options to set the state of the radio button as either checked or unchecked. When you select any option from the group of radio buttons, a dot mark appears inside the circle. This is called the **checked state**. Previously selected radio buttons in this group are unselected that is they go to the **unchecked state**. The **checked** property is used to set the state of the radio button.

The following steps explain the details about rendering the **RadioButton** with the **checked** option

In the **HTML** page, add the following input elements to configure **RadioButton** widget.

### HTML

```
<div class="page-align">
<table>
<tr>
<td>
<input type="radio" id="Radio_checked" />
</td>
<td>
<label for="Radio_checked" >Male</label>
</td>
</tr>
<tr>
<td>
<input type="radio" id="Radio_unchecked" />
</td>
<td>
<label for="Radio_unchecked">Female</label>
</td>
</tr>
</table>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RadioButtonComponent {
$(function () {
```

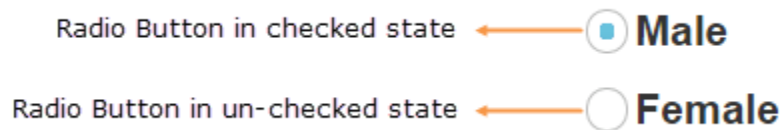
```
// Here we render checked and unchecked type of radio buttons in same group
// set checked state of radio button as follows
var radioButton = new ej.RadioButton($("#Radio_checked"), {
  name: "Gender",
  checked: true
});
var radioButton1 = new ej.RadioButton($("#Radio_unchecked"), {
  name: "Gender"
});
});
}
```

Configure the CSS styles to align the radio buttons.

### CSS

```
<style>
.page-align {
margin: 100px;
}
</style>
```

The following image is displayed as the output for the above steps.



### Text

Specifies the text content for the radio button. In previous programs, separate labels were created for each radio button. But now you have the option to set the text for radio button using the **text** property. So here you do not have to add a label tag for each radio button in the **HTML** code.

The following steps explain the details about rendering the **RadioButton** with **text** and without using the label tag options.

In the **HTML** page, add the following input elements to configure the **RadioButton** widget.

### HTML

```
<div class="page-align">
<table>
<tr>
<td>
<!--here we did not use label tag-->
<input type="radio" id="RadBtu_male" />
</td>
</tr>
<tr>
<td>
<!-- here we did not use label tag-->
<input type="radio" id="RadBtu_female" />
</td>
```

```

</tr>
</table>
</div>

```

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RadioButtonComponent {
$(function () {
// radio button with text property
var radioButton = new ej.RadioButton($("#RadBtu_male"), {
name: "Gender",
checked: true,
text: "Male"
});
var radioButton1 = new ej.RadioButton($("#RadBtu_female"), {
name: "Gender",
text: "Female"
});
});
}

```

Configure the CSS styles to align the radio buttons.

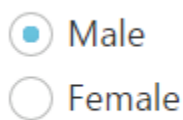
### CSS

```

<style>
.page-align {
margin: 100px;
}
</style>

```

The following image is displayed as the output for the above steps.



### Size

You can render the **RadioButton** in different sizes. There are some predefined size options available for rendering a **RadioButton** in an easy way. Each size option has different height and width. It mainly avoids the complexity in rendering **RadioButton** with complex CSS class.

#### Size

| Property | Description  |
|----------|--|
| small    | Creates radio button with Built-in small size height, width specified. |



|        |   |
|--------|---|
| medium | Creates radio button with Built-in medium size height, width specified. |
|--------|---|

The following steps explain the details about rendering **RadioButton** with different size options.

In the HTML page, add the following input elements to configure RadioButton widget.

### HTML

```
<div class="page-align">
<table>
<tr>
<td>Small size Radio buttons
</td>
</tr>
<tr>
<td>
<input type="radio" id="RadBtu_male" />
<label for="Radio_Male">Male</label>
</td>
</tr>
<tr>
<td>
<input type="radio" id="RadBtu_male" />
<label for="Radio_Female">Female</label>
</td>
</tr>
</table>
<table>
<tr>
<td>Medium size Radio buttons
</td>
</tr>
<tr>
<td>
<input type="radio" id="Radio1_Male" />
<label for="Radio1_Male">Male</label>
</td>
</tr>
<tr>
<td>
<input type="radio" id="Radio1_Female" />
<label for="Radio1_Female">Female</label>
</td>
</tr>
</table>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RadioButtonComponent {
$(function () {
// small size of radio buttons in same group
var radioButton = new ej.RadioButton($("#RadBtu_male"), {
```

```

name: "Gender",
size: "small",
checked: true
});
var radioButton1 = new ej.RadioButton($("#RadBtu_male"), {
name: "Gender",
size: "small"
});
// Medium size of radio buttons in same group
var radioButton2 = new ej.RadioButton($("#Radio1_Male"), {
name: "Gender1",
size: "medium",
checked: true
});
var radioButton3 = new ej.RadioButton($("#Radio1_Female"), {
name: "Gender1",
size: "medium"
});
});
}

```

Configure the CSS styles to align the radio buttons.

### CSS

```

<style>
.page-align {
margin: 100px;
}
</style>

```

The following image is displayed as the output for the above steps.

#### Small size Radio buttons

- ☒ **Male**
- ☐ **Female**

#### Medium size Radio buttons

- ☒ **Male**
- ☐ **Female**

### RTL Support

In some cases you need to use right-to-left alignment. You can give RTL support by using **enableRTL** property. **RTL** mode works when you use the **text** property in **RadioButton**. The **RadioButtons** and text are aligned in the right-to-left format. For example, when text is right-aligned and RadioButton is left-aligned, after you apply right-to-left alignment, these positions are interchanged.

The following steps explain the details about rendering the **RadioButton** with right-to-left alignment support. Here the **text** property is necessary.

In the HTML page, add the following button elements to configure RadioButton widget.

### HTML

```
<div class="page-align">
<table class="rightAlign">
<tr>
<td>
<input type="radio" id="RadBtu_male" />
</td>
</tr>
<tr>
<td>
<input type="radio" id="RadBtu_female" />
</td>
</tr>
</table>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RadioButtonComponent {
$(function () {
//set radio button with right to left format
var radioButton = new ej.RadioButton($("#RadBtu_male"), {
name: "Gender",
checked: true,
text: "Male",
enableRTL: true
});
var radioButton1 = new ej.RadioButton($("#RadBtu_female"), {
name: "Gender",
text: "Female",
enableRTL: true
});
});
}
```

In the above mentioned code, the **text** property has been used. In **LTR** format, the **RadioButton** is on the left side. In **RTL** format, the **RadioButton** appears on the right side. Here the **text** property is used and the **enableRTL** property is set as **"true"**. It changes the alignment to right-to-left.

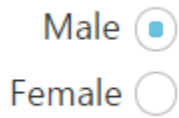
Configure the CSS styles to align the RadioButtons.

### CSS

```
<style>
.page-align {
margin: 100px;
}
.rightAlign {
```

```
text-align: right;
}
</style>
```

The following image is displayed as the output for the above steps.



## Miscellaneous

### RadioButton ID

**RadioButton id** is not shown in the user interface. Here id denotes the id attribute of the root element of **RadioButton** control. This id value is unique. You can give id through **element** and through the **id property**. When you use two id for a single radio button at initialization, the **element id** is considered.

Set **id** for **RadioButton** control as follows.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RadioButtonComponent {
  $(function () {
    //set new id value as follows
    var radioButton = new ej.RadioButton($("#RadBtn_male"), {
      name: "Gender",
      id: "male_type"
    });
    var radioButton = new ej.RadioButton($("#RadBtn_female"), {
      name: "Gender",
      id: "female_type"
    });
  });
}
```

### RadioButton Prefix id

Id prefix value is appended to id value. It is used to mention the prefix for the wrapper's id attribute. When you assign a value for **idPrefix** property, the older prefix id gets replaced by the new prefix id.

Setting a new prefix id for **RadioButton** control is as follows.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RadioButtonComponent {
  $(function () {
    //set new idPrefix value as follows
    var radioButton = new ej.RadioButton($("#RadBtn_male"), {
      name: "Gender",
      idPrefix: "sync"
    });
  });
}
```

```
var radioButton1 = new ej.RadioButton($("#RadBtn_female"), {
  name: "Gender",
  idPrefix:"sync"
});
});
}
```

### RadioButton Name

The **name** setting tells you where a **RadioButton** belongs. When you select one button, all other buttons in the same group are unselected. You can have only one group of **RadioButtons** on each page.

The **name** attribute is also used to identify form data after it has been submitted to the server, or for reference of form data using **JavaScript** on the client's side. Only form elements with a **name** attribute have their values passed, when submitting a form.

### RadioButton Value

The **value** setting defines what can be submitted when checked.

For **RadioButtons**, the contents of the value property do not appear in the user interface. The **value** property only has meaning when submitting a form. If a radio button is in the checked state when the form is submitted, the name of the Radio button is sent along with the value of the value property, if the radio button is not checked, no information is sent.

To identify, on the server side, which one was checked, give different values for radio buttons in the same group,

Set name and value for each radio button control as follows.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RadioButtonComponent {
  $(function () {
    //set name and value for each radio button as follows
    var radioButton = new ej.RadioButton($("#RadBtn_male"), {
      name: "Gender",
      value: "male"
    });
    var radioButton = new ej.RadioButton($("#RadBtn_female"), {
      name: "Gender",
      value: "female"
    });
  });
}
```

## RangeNavigator

### Overview

**RangeNavigator** is a data visualization control. It allows you to scroll and navigate through the data. This control easily combines with other controls such as Chart, GridView, etc., to create rich and powerful dashboards.

### Key Features

The **RangeNavigator** is composed of the following features:

- **Higher level label:** Contains a time span format one level higher than the lower level **DateTime** values. For example, if the higher level label contains the year format (yyyy), the lower level label contains the Quarter format.
- **Lower level label:** Contains a time span format one level lower than the higher level **DateTime** values. For example, if the lower level label contains the Quarter format, the higher level label contains the year format (yyyy).
- **Chart:** To represent the data in RangeNavigator.
- **Slider:** To handle the selected data in RangeNavigator.

## Getting Started

This section explains you the steps required to populate the RangeNavigator with data, add data labels, tooltips and title to the Chart. This section covers only the minimal features that you need to know to get started with the RangeNavigator.

### Create your RangeNavigator

This section encompasses on how to configure the ejRangeNavigator and update the chart control for RangeNavigator's selected range. It also helps you to learn how to pass the required data to RangeNavigator and customize the scale and selected range for your requirements. In this example, you will look at the steps to configure a RangeNavigator to analyze sales of a product for a particular quarter in a year.



### Configure ejRangeNavigator

Getting started with your ejRangeNavigator is simple. You can initialize the ejRangeNavigator by setting its range values.

1.First create an TypeScript Project and the following script reference in the app.ts file

For common getting started of TypeScript , you can refer [here](#).

The default type definition file **ej.web.all.d.ts** needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the **jquery.d.ts** file in your TypeScript application, which can be downloaded from [here](#).

2.Add the below script reference in the HTML page

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

In the above code, **ej.web.all.min.js** script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

3.Create a

tag.

#### HTML

```
<html> <body> <div id="RangeNavigator"></div> </body> </html>
```

4.Initialize the RangeNavigator in ts file by using the **ej.RangeNavigator** method.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RangeNavigatorComponent {
$(function () {
```

```

var rangeNavigatorSample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
  rangeSettings: {
    start: "2010/1/1", end: "2010/12/31"
  },
});

```

The following Screen shot displays the RangeNavigator with a range from 2010 January 1st to December 31st.



### Add series

To add a series to **RangeNavigator**, you need to set **dataSource** property, as given in the following code example.

You can create data source for RangeNavigator as follows.

### JAVASCRIPT

```

var data = [{ "xDate": new Date(2011, 0, 1), "yValue": 10 },
{ "xDate": new Date(2011, 2, 1), "yValue": 5 },
{ "xDate": new Date(2011, 4, 1), "yValue": 15 },
{ "xDate": new Date(2011, 6, 1), "yValue": 25 },
{ "xDate": new Date(2011, 8, 1), "yValue": 10 },
{ "xDate": new Date(2011, 10, 1), "yValue": 5 },
{ "xDate": new Date(2011, 12, 1), "yValue": 15 }];

```

Now, add the dataSource to the RangeNavigator and provide the field name to get the values from the dataSource in xName and yName options

### JAVASCRIPT

```

$(function () {
  var rangeNavigatorSample = new
  ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
    series: [
      {
        type: 'line',
        dataSource: data, xName: "XValue", yName: "YValue",
      }
    ]
  });
});

```

The following screenshot displays the RangeNavigator with the type series as “line”.





### Enable tooltip

Tooltip can be customized for RangeNavigator using tooltip option. You can also use TooltipDisplayMode option in tooltip to display the tooltip “always” or “ondemand” (displays tooltip only while dragging the sliders). You can also specify label format for tooltip using LabelFormat.

### JAVASCRIPT

```
$(function () {
    var rangeNavigatorSample = new
    ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
        tooltipSettings: {
            visible: true, labelFormat: "MMM/yyyy", tooltipDisplayMode: "always"
        },
    });
});
```

The following screen shot displays the label format Tooltip in RangeNavigator:



### Update Chart

You can use ejRangeNavigator with controls such as chart and grid to view the range of data selected in ejRangeNavigator.

In order to update chart, whenever the selected range changes in ejRangeNavigator, you need to use rangeChanged event of ejRangeNavigator and then update the chart with the selected data in this event.

You can create a chart with line series using the following code sample.

Create a <div> tag with an id for rendering the chart.

### HTML

```
<body>
<div id=" Chart "></div>
</body>
```

### JAVASCRIPT

```
$(function () {
    var rangeNavigatorSample = new
    ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
        dataSource: data, xName: "XValue", yName: "YValue",
```

```

rangeChanged: function (sender) {
var chartObj = $("#Chart").data("ejChart");
if (chartObj != null) {
chartObj.model.series[0].dataSource = sender.selectedData;
$("#Chart").ejChart("redraw");
}
}
});
var chart = new ej.datavisualization.Chart($("#Chart"), {
title: { text: "Sales Analysis" },
legend: { visible: true, position: 'top' },
primaryYAxis: {
title: { text: "Sales(Million)" }
},
series: [
{
name: 'Product A', type: 'line',
dataSource: data, xName: "xDate", yName: "yValue"
}
],
});
});

```

The following screenshot displays how the RangeNavigator is updated when the selected range is changed.



### Set value type

RangeNavigator can also be used with numerical values. You can specify the data type using ValueType option.

First let's create a DataSource for Chart Series with integer Values.

### JAVASCRIPT

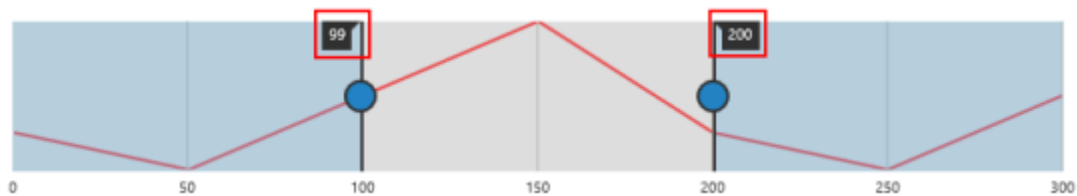
```
var Data = [
  { "xDate": 0, "yValue": 10 },
  { "xDate": 50, "yValue": 5 },
  { "xDate": 100, "yValue": 15 },
  { "xDate": 150, "yValue": 25 },
  { "xDate": 200, "yValue": 10 },
  { "xDate": 250, "yValue": 5 },
  { "xDate": 300, "yValue": 15 },
];
```

Now, you can set the dataSource for Chart Series and valueType property to "numeric" as given in the following code example.

### JAVASCRIPT

```
$(function () {
  var rangeNavigatorSample = new
  ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
    series: [
      {
        type: 'line',
        dataSource: data, xName: "XValue", yName: "YValue",
      }
    ],
    valueType: "numeric"
  });
});
```

The following screenshot displays the RangeNavigator with numerical values:



### Range Types

**RangeNavigator** control is designed to visualize large number of data and navigate to particular data from the large collection at ease. The data for the **RangeNavigator** is either numeric values or **DateTime** values and the **valueType** property in **RangeNavigator** indicates the type of the data that should be passed for the control. By default the **valueType** of **RangeNavigator** is **DateTime**

- Numeric
- DateTime

### Numeric Type

**RangeNavigator** is also used with numeric data and the **valueType** for this data is "numeric"

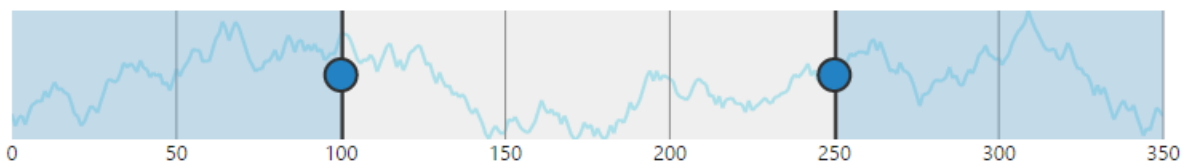
### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module RangeComponent {
$(function () {
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
//...
valueType: "numeric",
//...
});
});
}

```

The following screenshot displays the **RangeNavigator** with numeric data.



### DateTime

By default the **valueType** of the **RangeNavigator** is **"datetime"** and represents the **DateTime** values.

### JAVASCRIPT

```

var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
//...
valueType: "datetime",
//...
});

```



### DateTime Intervals

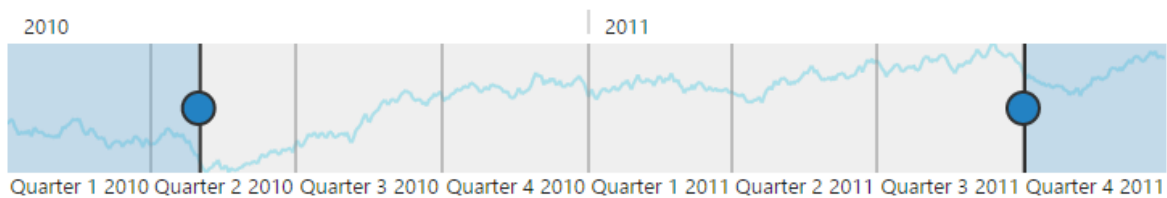
The **DateTime** range type contains an **intervalType** property that sets the **DateTime** interval to one of the following:

- Years
- Quarters
- Months
- Weeks
- Days
- Hours

By default **intervalType** for higher level labels are **years** and for lower level labels its **quarters**.

### JAVASCRIPT

```
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
//...
labelSettings:
{
higherLevel:
{
intervalType: 'years',
},
lowerLevel:
{
intervalType: 'quarters',
}
},
//...
});
```



### Range Padding

**Range Padding** adds padding for range in **RangeNavigator**. It allows you to space the grid lines in the **RangeNavigator**. By default, this property is set to **none**.

### Numeric

The **rangePadding** property allows you to customize the automatic range calculation using the default auto range calculation for **RangeNavigator**.

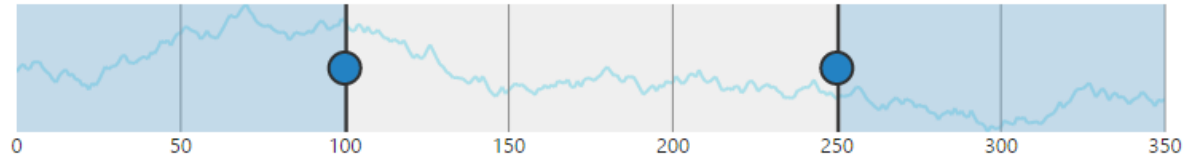
### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module RangeComponent {
$(function () {
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
//...
valueType: "numeric",
rangePadding: 'none ',
//...
});
});
}
```

### None

By default, the **rangePadding** for numerical range is none. The range is calculated from the minimum value to the maximum value of data in the RangeNavigator.

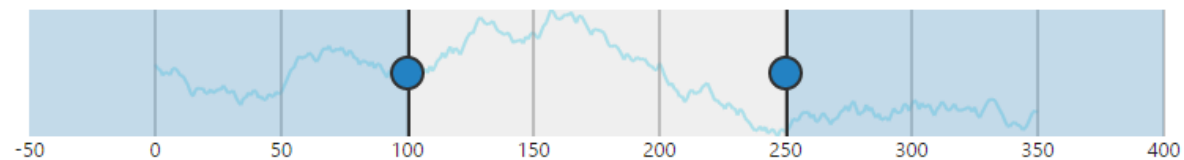
The following screenshot illustrates a **RangeNavigator** with **rangePadding** set to none.



### Additional

When you set the **rangePadding** for numerical range to **Additional**, range is padded with an interval.

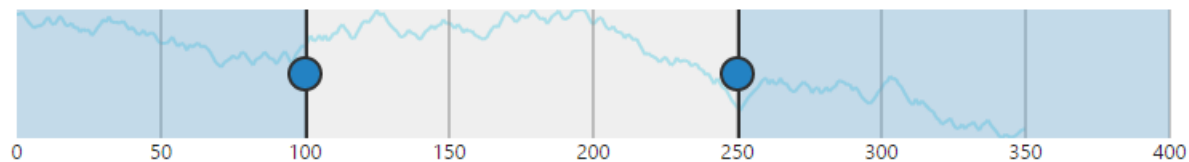
The following screenshot illustrates a **RangeNavigator** with **rangePadding** set to additional.



### Normal

In normal **rangePadding**, automatic range calculation differs based on the data.

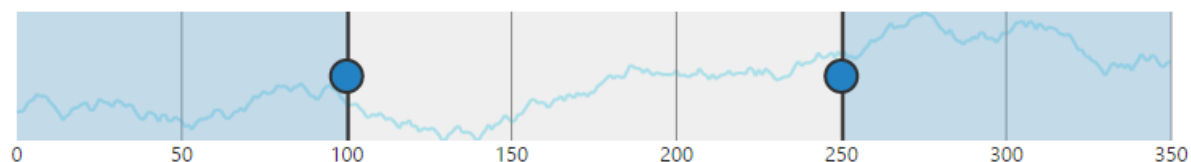
The following screenshot illustrates **RangeNavigator** with **rangePadding** set to normal



### Round

Round **rangePadding** for a numerical range rounds the range of the control to the nearest possible value that is divisible by the interval.

The following screenshot illustrates a **RangeNavigator** with **rangePadding** set to **Round**.



### DateTime

Using the default range calculation for **RangeNavigator**, the **rangePadding** property allows you to customize the range.

## JAVASCRIPT

```
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
//...
rangePadding: 'none ',
//...
});
```

### None

By default, the **rangePadding** for **DateTime** range is none. The range is calculated from the minimum value to the maximum value of data in the RangeNavigator

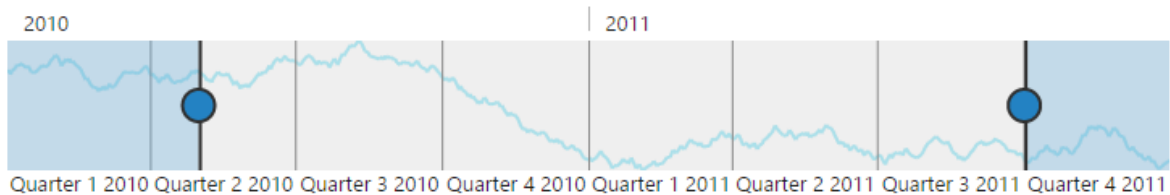
The following screenshot illustrates a **RangeNavigator** with **rangePadding** set to none.



### Round

Round **rangePadding** for a **DateTime** range rounds the range of the control to the nearest possible value.

The following screenshot illustrates a **RangeNavigator** with **rangePadding** set to Round.



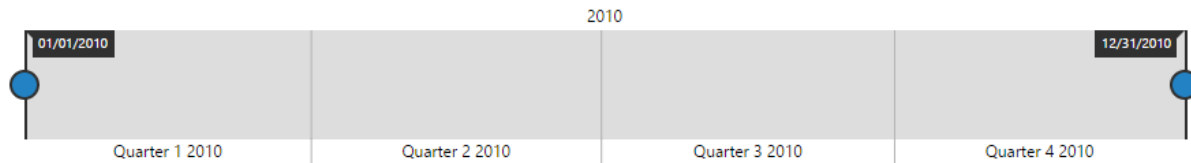
### Customize axis range of navigator

**RangeNavigator** calculates the range automatically based on the values of series data points. However you can explicitly specify the range using the **start**, **end** properties in **rangeSettings** that is not possible when data is provided.

The following code example renders a RangeNavigator with a range from 2010 January 1st to 2013 January 1st.

### JAVASCRIPT

```
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
//...
rangeSettings: {
start: "2010/1/1", end: "2012/13/1"
},
//...
});
```



### Populate Data

When you provide data to **RangeNavigator**, it produces limited set of data. You can populate the **RangeNavigator** with data using the **dataSource** and **series** properties.

### Add series to the RangeNavigator

The **Series** property provides access to a collection of all series that are defined explicitly within a **RangeNavigator** control. Each series is assigned with type and name. It contains collection of data point, each point contains x value and y values. You can add data points to the series through **dataSource** property.

### JAVASCRIPT

```

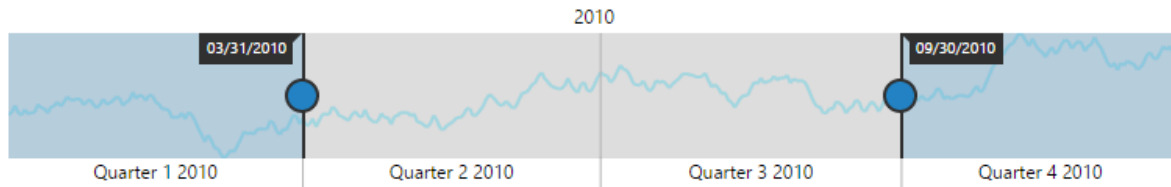
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module RangeComponent {
$(function () {
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
//...
// Adding series
series: [
{
type: 'line',
dataSource: data.Open, xName: "XValue", yName: "YValue",
opacity: 0.5, fill: '#69D2E7',
border: { color: 'transparent', width: 2 }
},
],
//...
});
});
}

var data;
function GetData() {
var series1 = [];
var value = 100;
for (var i = 1; i < 360; i++) {
if (Math.random() > .5) {
value += Math.random();
} else {
value -= Math.random();
}
var point1 = { XValue: new Date(2010, 0, i), YValue: value };
series1.push(point1);
}
data = { Open: series1};
return data;
}
}

```



The following screenshot illustrates the **RangeNavigator** that is populated with data using **dataSource** property in series.



### Behavior Customization

**RangeNavigator** allows you to customize the control using events. You can change the range for selected data of the **RangeNavigator** using events.

#### Deferred update

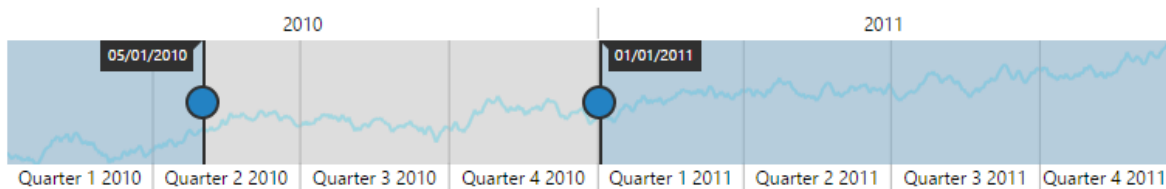
If you set **enableDeferredUpdate** to true, the **rangeChanged** event gets fired after dragging and dropping the slider. By default the **enableDeferredUpdate** is true. If **enableDeferredUpdate** is false then the **rangeChanged** event gets fired while dragging the slider.

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module RangeComponent {
  $(function () {
    var sample = new
    ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
      //...
      enableDeferredUpdate: true,
      //...
    });
  });
}

```



### Handle Events

#### loaded: function

This event is handled when the **RangeNavigator** gets loaded. A parameter **sender** is passed to the handler. Using **sender.model**, you can access the **RangeNavigator** properties.

### JAVASCRIPT

```

var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
  // ...
  loaded: function(sender) {
    sender.model.isResponsive = false;
  },
}

```

```
//...  
});  
});
```

**rangeChanged:** function

This event gets fired whenever the selected range changes in **RangeNavigator**. A parameter **sender** is passed to the handler. Using `sender.selectedRangeSettings`, you can access the start and end value of range for the selected region.

**JAVASCRIPT**

```
var sample = new  
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {  
  // ...  
  rangeChanged: function(sender) {  
    console.log(sender.selectedRangeSettings.start);  
  },  
  // ...  
});  
});
```

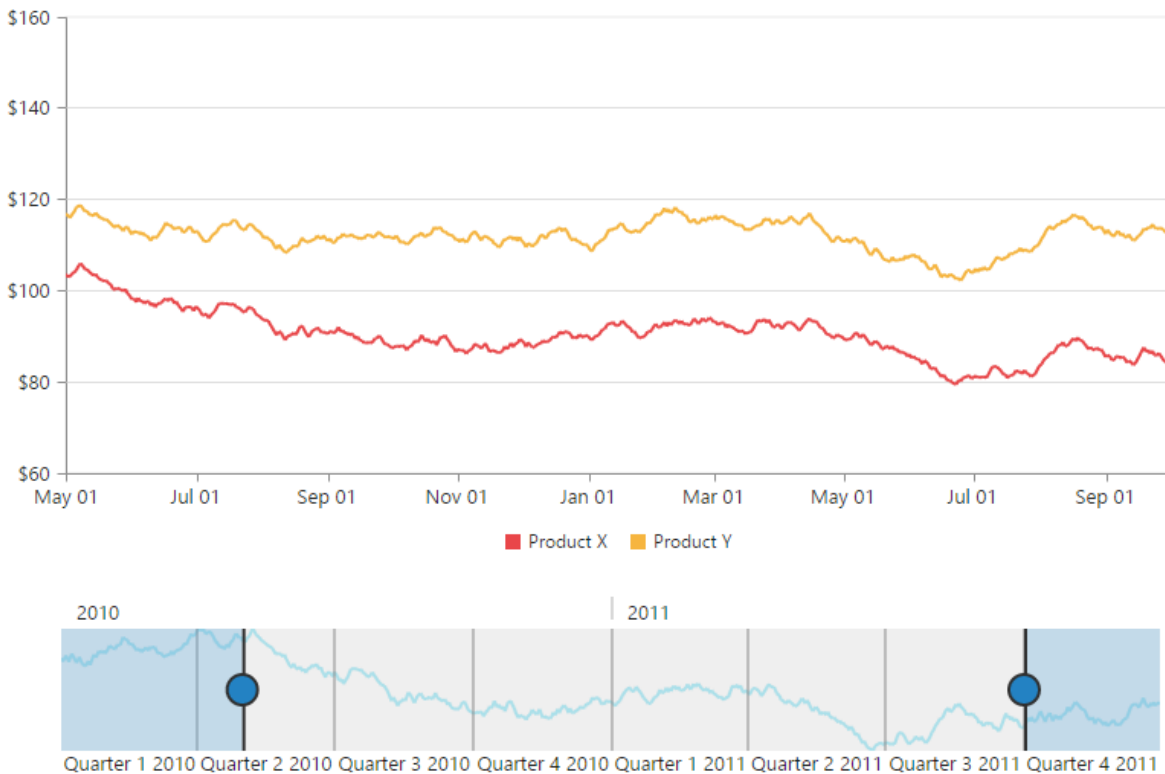
[Use of ZoomCoordinates](#)

**RangeNavigator** is used along with the controls like chart and grid to view the selected data. To update chart/grid, whenever the selected range changes in **RangeNavigator**, you can use **rangeChanged** event of **RangeNavigator** and then update the chart/grid with the selected data in this event.

You can easily update the data for chart by assigning the **zoomFactor** and **zoomPosition** of the **RangeNavigator** to the chart axis.

**JAVASCRIPT**

```
var sample = new  
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {  
  // ...  
  rangeChanged: onChartLoaded  
  // ...  
});  
});  
  
// setting zoom factor and position for chart axis in rangeChanged event.  
function onChartLoaded(sender) {  
  var chartObj = $("#container").data("ejChart");  
  if (chartObj != null) {  
    chartObj.model.axes[0].zoomPosition = sender.zoomPosition;  
    chartObj.model.axes[0].zoomFactor = sender.zoomFactor;  
  }  
  $("#container").ejChart("redraw");  
}
```



### Thumb Template

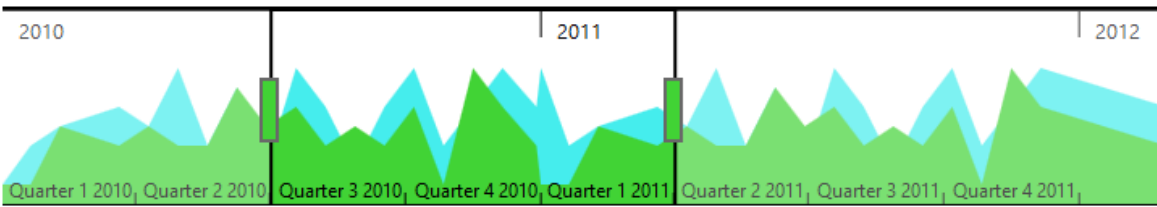
You can customize Thumb template by using **leftThumbTemplate** and **rightThumbTemplate** property. You can add the required template as a "div" element with an "id" to the web page and assign the id or assign the HTML string to this property under **navigatorStyleSettings**.

### HTML

```
<script type="text/x-jsrender" id="left" >
<svg height="24" width="32" style="fill:#DD4A4A;stroke:black;">
<path d="M2 2 L2 22 L22 22 L32 12 L22 2 Z" />
</svg>
</script>
<script type="text/x-jsrender" id="right">
<svg height="24" width="32" style="fill:#DD4A4A;stroke:black; ">
<path d="M2 12 L12 22 L32 22 L32 2 L12 2 Z" />
</svg>
</script>
<script type="text/javascript" language="javascript">
$(function () {
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
// ...
navigatorStyleSettings: {
leftThumbTemplate: 'left',
rightThumbTemplate: 'right',
},
// ...
});
});
```

```
</script>
```

The following screenshot displays the **RangeNavigator** using thumb template.



### Appearance and Styling

**RangeNavigator** is enriched with lots of customization options for labels, gridlines and slider to develop high quality graphic rich control.

#### Customize labels

The labels are found along the range, displaying the value of the data it correspond, both on (higher level label) and below (lower level label) the **RangeNavigator**. **RangeNavigator** labels are further customized using "font" property in label Settings.

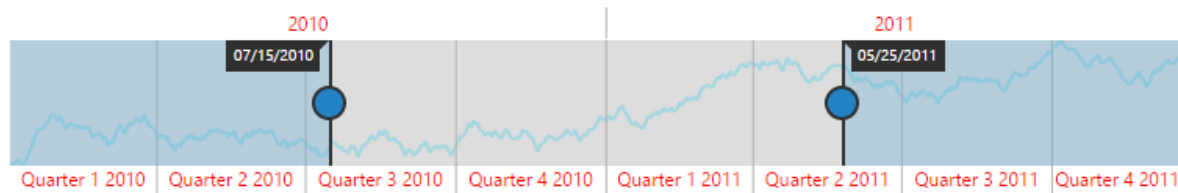
### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module RangeComponent {
$(function () {
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
// ...
labelSettings: {
// customizing higher level labels.
higherLevel: {
style: {
font: {
color: '#ff0000',
style: 'Normal',
size: '12px',
opacity: 1,
weight: 'regular'
},
},
},
// customizing lower level labels.
lowerLevel: {
style: {
font: {
color: '#ff0000',
style: 'Normal',
size: '12px',
opacity: 1,
weight: 'regular'
},
},
},
}
```

```

},
// ...
});
});
}

```



### Label Placement

Labels in **RangeNavigator** are placed inside or outside of the control. You can customize both the higher and lower level labels using **labelPlacement** property in label setting of **RangeNavigator**. By default **labelPlacement** is "outside" for the both higher and lower level labels.

The following screen shot illustrates both the lower and higher level labels that are placed outside the control with **labelPlacement** specified as outside.

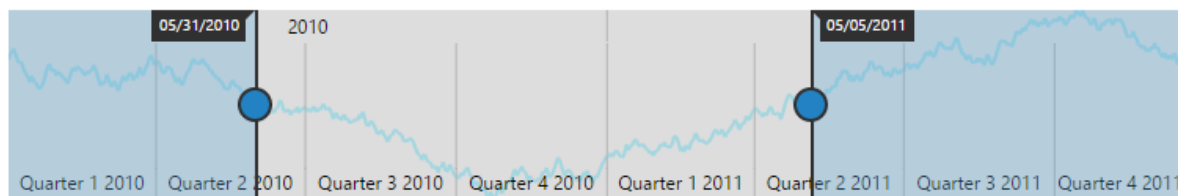
### JAVASCRIPT

```

var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
// ...
labelSettings: {
higherLevel: {
labelPlacement: "inside",
},
lowerLevel: {
labelPlacement: "inside"
}
},
// ...
});
});

```

The following screenshot illustrates a **RangeNavigator** with labels inside the control after specifying the **labelPlacement** as inside.



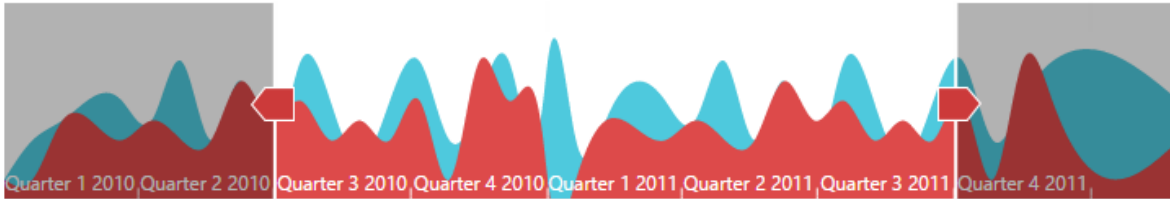
### Customize RangeNavigator

RangeNavigator is customized using **navigatorStyleSettings** properties. You can customize the selected and unselected region color using **selectedRegionColor**, **unselectedRegionColor** in **navigatorStyleSettings** and the thumb of the slider using **thumbColor**, **thumbRadius** and **thumbStroke**

in **navigatorStyleSettings**. **majorGridLineStyle** and **minorGridLineStyle** are used to customize the grid line color and visibility.

### JAVASCRIPT

```
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
// ...
// To customize the navigator element
navigatorStyleSettings: {
unselectedRegionColor: "white",
selectedRegionColor: "#5EABDE",
thumbColor: "white",
thumbRadius: 10,
thumbStroke: "#303030",
background: "transparent",
border: {
color: "black",
width: 3
},
majorGridLineStyle: {
color: "transparent"
},
minorGridLineStyle: {
color: "transparent"
}
},
// To customize the labels
labelSettings: {
higherLevel: {
style: {
font: {
color: 'black',
size: '13px',
opacity: 1
},
horizontalAlignment: "left"
},
intervalType: 'years',
labelPlacement: "inside"
},
lowerLevel: {
style: {
font: {
color: 'black',
size: '12px',
opacity: 1
},
horizontalAlignment: "center"
},
intervalType: 'quarters',
labelPlacement: "inside"
}
},
// ...
});
});
```



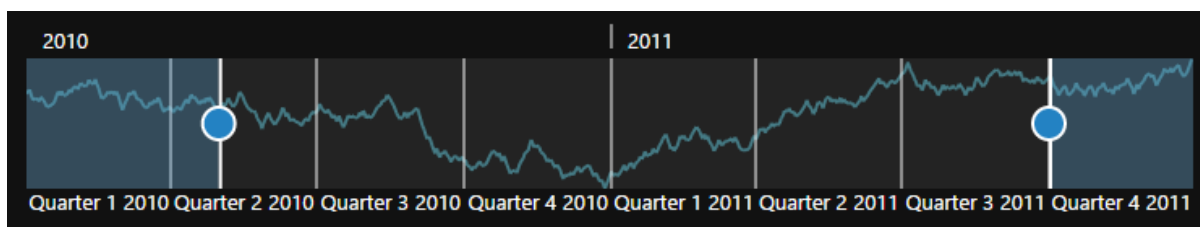
### Themes

**RangeNavigator** theme is a set of pre-defined options that are applied to the control before each **RangeNavigator** is instantiated. Following predefined themes are available in JavaScript **RangeNavigator**.

1. flat light
2. flat dark
3. gradient light
4. gradient dark
5. azure
6. azure dark
7. lime
8. lime dark
9. saffron
10. saffron dark
11. gradient azure
12. gradient azure dark
13. gradient lime
14. gradient lime dark
15. gradient saffron
16. gradient saffron dark

### JAVASCRIPT

```
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
// ...
theme: 'saffron',
// ...
});
```



### Tooltip

**RangeNavigator** provides **Tooltip** support for sliders. Sliders are used to select data at particular range in the **RangeNavigator** control. **Tooltips** for sliders display the selected start and end **DateTime** values.

### Customization

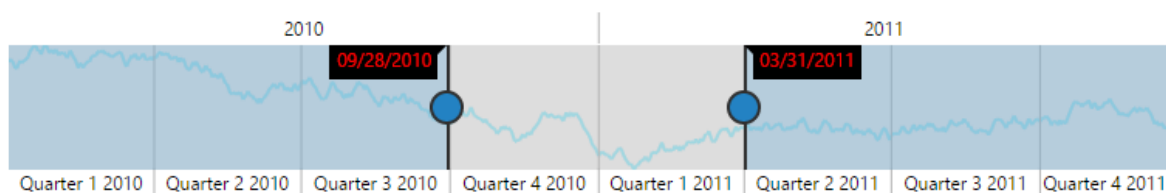
**RangeNavigator** provides support for you to customize the text display in the tooltip and background using **tooltipSettings** property. You can change font family, font color, font style, font weight. By default "Segoe UI" font family is set to tooltip text.

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module RangeComponent {
$(function () {
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
// ...
tooltipSettings: {
visible: true,
backgroundColor: "black",
// To customize the tooltip text
font: {
color: 'red',
family: 'Segoe UI',
style: 'Normal',
size: '12px',
opacity: 1,
weight: 'regular'
}
},
// ...
});
});
}

```



### Label Format

By default, the **tooltip** texts are automatically determined based on the data points. To make it readable and understandable you can format the **tooltip** text. For **DateTime** data, all globalized format are supported. By default the **labelFormat** is "MM/dd/yyyy".

Some of the **labelFormat** for **DateTime** data area as follows:

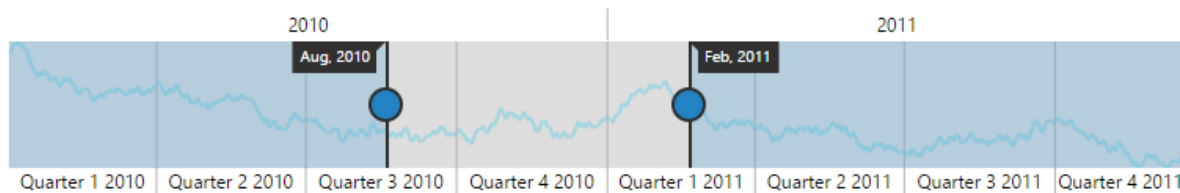
- 'MMM, yyyy'
- 'dd, MMM'
- 'dd/MM/yyyy'
- 'dd, hh:mm'



- 'hh:mm:ss'
- 'hh:mm:ss:tt'

### JAVASCRIPT

```
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
// ...
tooltipSettings: {
labelFormat: 'MMM, yyyy',
},
// ...
});
```



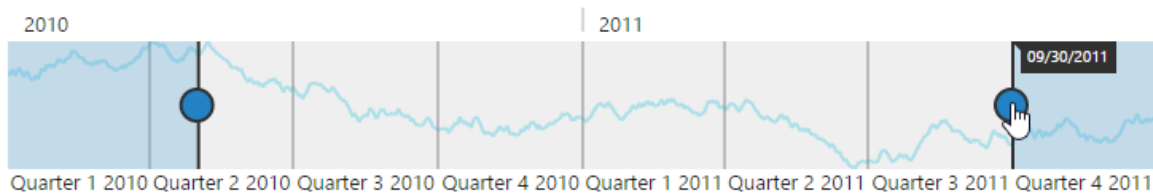
### Tooltip display mode

By default the **tooltip** for RangeNavigator gets displayed. You can change this behavior using the **tooltipDisplayMode** property in the tooltip and it takes the following values.

| Value    | Description   |
|----------|---|
| always   | Tooltip get displayed for RangeNavigator always.    |
| onDemand | Tooltip get displayed only when we move the slider. |

### JAVASCRIPT

```
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
// ...
tooltipSettings: {
tooltipDisplayMode: "onDemand",
},
// ...
});
```



### Globalization & Localization

**RangeNavigator** supports Localization and Globalization to customize the labels based on a culture specific to a country. The culture defines specific information for the number formats, week and month names, date and time formats etc.

#### Localization

**Localization** is the process of customizing the user interface based on a culture specific to a particular country or region in order to display regional data. The culture is represented by a unique string, for example, —en-US|| for U.S. English and —fr-FR|| for French (common), this is achieved by creating a JavaScript file “**rangeNavigatorSource.fr-FR.js**” and setting the equivalent word as illustrated in the following code sample.

#### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module RangeComponent {
$(function () {
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
locale: 'fr-Fr',
intervals: {
//string to display the intervals on RangeNavigator
quarter: {longQuarters: "Trimestre", shortQuarters: "T"},
week: { longWeeks: "Semaine", shortWeeks: "S" }
}
});
});
}

```

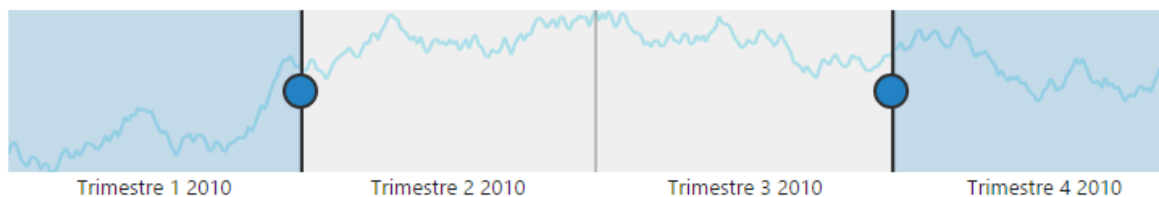
**Localization** is the key feature that provides solutions globally with the help of localized control.

#### JAVASCRIPT

```

var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
// ...
locale: 'fr-FR',
// ...
});

```



#### RTL

**Right-to-Left** or **RTL** describes the ability of application to handle and responds you to communicate with a right-to-left language, like Arabic or Japanese. **enableRTL** property is used to change the rendering format to “**Right to Left**”, by default it renders from “**Left to Right**” in **RangeNavigator**.

#### JAVASCRIPT

```

var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
// ...
enableRTL:true,
// ...
});

```



## User Interactions

### Highlight

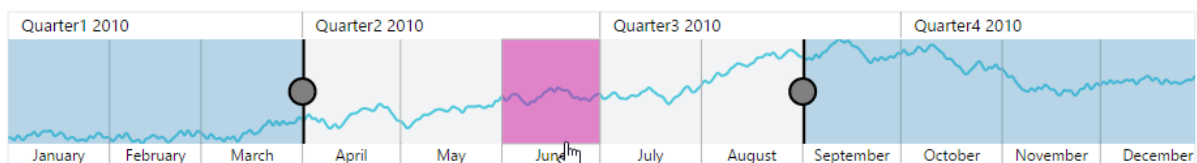
EjRangeNavigator provides highlighting supports to the intervals on mouse hover. To enable the highlighting option, set the `enable` property to true in the `highlightSettings` of `navigatorStyleSettings`.

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module RangeComponent {
$(function () {
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
//...
navigatorStyleSettings:{
//...
highlightSettings:{
// enable the highlight settings
enable: true
}
//...
}
// ...
});
});
}

```



[Click](#) here to view the highlight and selections online demo sample.

### Customize the highlight style

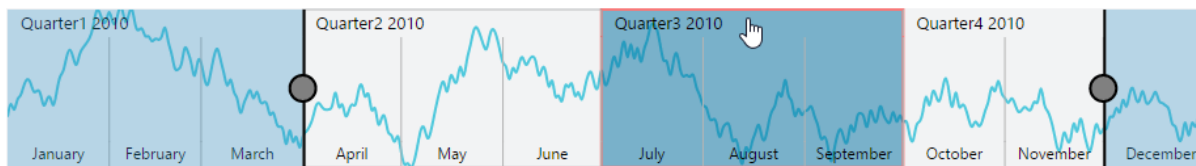
To customize the highlighted intervals, use color, border and opacity options in the `highlightSettings`.

**JAVASCRIPT**

```

var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
//...
navigatorStyleSettings:{
//...
highlightSettings:{
// enable the highlight settings
enable: true,
// customizing style
color: '#006fa0',
border:{
color: 'red' , width: 2
}
}
//...
}
// ...
});

```

**Selection**

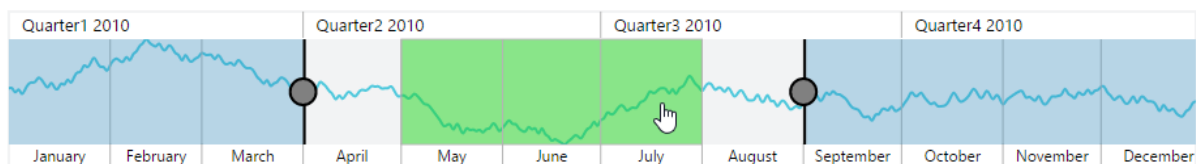
EjRangeNavigator provides selection supports to the intervals by, clicking and dragging the highlighted intervals. To enable the selection option, set the `enable` property to true in the `selectionSettings`.

**JAVASCRIPT**

```

var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
//...
navigatorStyleSettings:{
//...
selectionSettings:{
// enable the selection settings
enable: true
}
//...
}
// ...
});

```



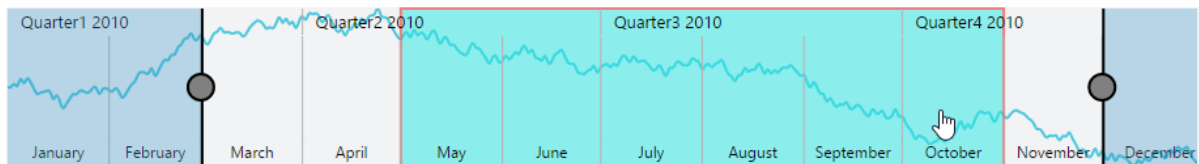
[Click](#) here to view the highlight and selections online demo sample.

*Customize the selection style*

To customize the selected intervals, use color, border and opacity options in the selectionSettings.

**JAVASCRIPT**

```
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
//...
navigatorStyleSettings:{
//...
selectionSettings: {
// enable the selection settings
enable: true,
// customizing style
color: '#27e8e5',
border:{
color: 'red' , width: 2
}
//...
}
// ...
});
```

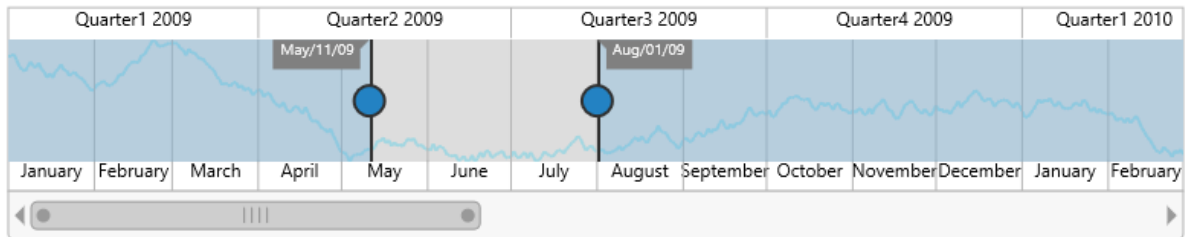
**Scrollbar**

- To render the Scrollbar in RangeNavigator, you need to enable `enableScrollbar` option.
- `scrollRangeSettings` of range navigator `start` and `end` value is used to set the minimum and maximum datasource value to be added in the range navigator.
- Based on the `scrollRangeSettings start, end` value and `dataSource start, end` value scrollbar will be adjust.
- When you change the scrollbar position, `scrollEnd` event returns the current position of start and end range value.

**JAVASCRIPT**

```
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
//...
//Enable scrollbar option in the range navigator
enableScrollbar: true,
//Maximum data to be displayed in the range navigator control
scrollRangeSettings: {
start: new Date(2010, 0, 1),
end: new Date(2011, 10, 31)
},
//Subscribe the event on scrollbar position changed
scrollEnd: "onScrollbarChange"
//...
```

```
});
function onScrollbarChange(sender) {
var start = sender.data.newRange.start;
var end = sender.data.newRange.end;
}
```



[Click](#) here to view scrollbar online demo sample.

## How to

### Change the default type of the series

The **type** property in **series** is used to change the type of the series rendered within **RangeNavigator**. By default line series will be rendered to represent the data in **RangeNavigator**.

### JAVASCRIPT

```
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
{
// ...
series: [{
type: 'column',
// ...
}],
// ...
});
```



When using multiple series in **RangeNavigator**, **type** property in **seriesSettings** is used to set a type common for all the series.

### JAVASCRIPT

```
var sample = new
ej.datavisualization.RangeNavigator($("#RangeNavigator"), {
{
// Using a common series type for all the series
seriesSettings: {
type: 'StackingColumn',
// ...
},
// ...
});
```



## Rating

### Overview

Essential **JavaScriptRating** control provides an intuitive **Rating** experience that enables you to select a number of stars that represent a Rating. You can configure the item size, orientation and the number of displayed items in the **Rating** control. You can also customize the **Rating** star image by using custom CSS.

### Key Features

Some important features of Chart for Essential **JavaScript** are as follows:

- **Precision support** - You can set the three different precisions like full, half, and exact.
- **Orientation support** - Rating control can be displayed in horizontal or vertical direction.
- **Read-only mode** - Rating control can be rendered in the read-only mode to enable or disable user interaction.
- **Reset** - You can reset the Rating value by using the Reset button.
- **Tooltip** - You can set a tooltip for displaying the currently selected Rating value.
- **Theme** - Rating control consists of 17 built-in themes (6 flat, 6 gradient, bootstrap, 2 high-contrast, material and office-365 effects) and also support the custom skin option to set user-defined themes.

### Getting Started

This section explains briefly about how to create a **Rating** control in your application with **JavaScript**. **Essential JavaScript Rating** helps to select the number of stars that represent Rating. Here, you can learn how to create **Rating** control in a real-time movie download application and also learn how to rate the application.

The following screenshot illustrates the functionality of a Rating widget with a Rating range of 0 to 5.



### Create a Rating Widget

**Essential JavaScript Rating** widget is provided with built-in features such as precision, orientation and flexible API's. You can create the **Rating** widget by using input textbox element as follows:

Create an HTML file and add the following template to the HTML file.

### HTML

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>Getting Started - RichTextEditor</title>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"></script>
</head>
<body>
<!-- add rating element here -->
</body>
```

```
</html>
```

Add movies list in the table to render Rating control. Here, you can add corresponding images in the images folder and give the accurate image path.

### HTML

```
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<div style="width: 500px">
<div id="moviesTab">
<ul>
<li><a href="#steelman">Man of Steel</a></li>
<li><a href="#woldwar">World War Z</a></li>
<li><a href="#unive">Monsters University</a></li>
</ul>
<div id="steelman">
<div>
<table>
<tr>
<td class="movies-img" valign="top">

</td>
<td valign="top">
<div>
<span class="movie-header">Man of Steel</span><br />
Rating :
<br />
<input id="mosRating" type="text" class="rating" />
<span>Movie Info:</span>
<p>
A young boy learns that he has extraordinary powers and is not of this
Earth.
</p>
</div>
</td>
</tr>
</table>
</div>
</div>
<div id="woldwar">
<table>
<tr>
<td class="movies-img" valign="top">

</td>
<td valign="top">
<div>
<span class="movie-header">World War Z</span><br />
Rating :
<br />
<input id="wwzRating" type="text" class="rating" />
<span>Movie Info:</span>
<p>
The story revolves around United Nations employee Gerry Lane (Pitt).
```



```

</p>
</div>
</td>
</tr>
</table>
</div>
<div id="unive">
<table>
<tr>
<td class="movies-img" valign="top">

</td>
<td valign="top">
<div>
<span class="movie-header">Monsters University</span><br />
Rating :
<br />
<input id="univRating" type="text" class="rating" />
<span>Movie Info:</span>
<p>
Mike Wazowski and James P. Sullivan are an inseparable pair, but that
wasn't always the case.
</p>
</div>
</td>
</tr>
</table>
</div>
</div>
</div>
</div>
</div>
</div>

```

Initialize the Rating in ts file by using the `ej.Rating` method.

### JAVASCRIPT

```

/// <reference path="jquery.d.ts" />
/// <reference path="scripts/typings/ej/ej.web.all.d.ts" />
module RatingComponent {
$(function () {
var tab = new ej.Tab($("#moviesTab"));
var rating = new ej.Rating($("#mosRating"), {
value: 2
});
var rating1 = new ej.Rating($("#wwzRating"), {
value: 4
});
var rating2 = new ej.Rating($("#univRating"), {
value: 4
});
});
});
}

```

Apply the following styles to show the Rating widget in the horizontal order.

### CSS

```
<style type="text/css" class="cssStyles">
.movies-img {
width: 125px;
}
.movie-header {
font-size: 20px;
font-weight: 600;
}
</style>
```

The following screenshot displays a Rating widget.

![[/js/Rating/Getting-Startedimages/Getting-Startedimg2.png]

### Set the Min and Max Value

In a real-time scenario, you can extend the range by using the properties **minValue** and **maxValue** in the **Rating** widget.

### HTML

```
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<div class="frame">
<table>
<tr>
<td valign="top">Rate:
</td>
<td>
<input id="fullRating" type="text" class="rating" />
</td>
</tr>
</table>
</div>
</div>
</div>
</div>
```

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="scripts/typings/ej/ej.web.all.d.ts" />
module RatingComponent {
$(function () {
var rating = new ej.Rating($("#mosRating"), {
value: 2,
minValue: 2,
maxValue: 10
});
});
}
```

The above code example displays the following output.



### Set Precision

In a real-time movie **Rating** scenario, you can set precision in between two whole numbers, such as 2.5 or 3.7 and it is done using the property **precision** by changing the value to **ej.Rating.Precision.Half** or **ej.Rating.Precision.Exact**.

### HTML

```
<body>
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<div class="frame">
<table>
<tr>
<td valign="top">Full Precision :
</td>
<td>
<input id="fullRating" type="text" class="rating" />
</td>
</tr>
<tr>
<td valign="top">Half Precision :
</td>
<td>
<input id="halfRating" type="text" class="rating" />
</td>
</tr>
<tr>
<td valign="top">Exact Precision :
</td>
<td>
<input id="exactRating" type="text" class="rating" />
</td>
</tr>
</table>
</div>
</div>
</div>
</div>
```

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="scripts/typings/ej/ej.web.all.d.ts" />
module RatingComponent {
$(function () {
var fullRating = new ej.Rating($("#fullRating"), {
value: 4,
});
var halfRating = new ej.Rating($("#halfRating"), {
precision: ej.Rating.Precision.Half,
value: 3.5
});
});
}
```

```

var exactRating = new ej.Rating($("#exactRating"), {
  precision: ej.Rating.Precision.Exact,
  value: 3.7
});
});
}

```

The above code example displays the following output.



You can also add additional functionalities such as orientation, event tracer and API's to the **Rating**.

## Rating Customization

### Setting Value

The **Value** property sets the display value of the Rating. For example, when the **Value** property is set to be **four**, the Rating control renders four ratings. By default, **Value** property's value is **one**.

The following code example is used to render the **Rating** control with customized value.

Add the following HTML to render Rating with customized value.

### HTML

```

<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
  <table>
    <tr>
      <td valign="top">Rating:
    </td>
    <td>
      <input id="rating" type="text" />
    </td>
    </tr>
  </table>
</div>

```

### JAVASCRIPT

```

// Add the following script to render Rating with customized value.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RatingComponent {
  $(function () {
    var sample = new ej.Rating($("#rating"), {
      value: 4
    });
  });
}

```

The following screenshot illustrates the **Rating** with custom defined value.



### Min Value

**EJ Rating** control provides support for setting **minimum value**. This is achieved by adding `minValue` property. When the `minValue` property is set, the Rating value starts with **minValue+1**.

The following code example is used to render the **Rating** control with **minimum rating**.

Add the following HTML to render Rating with minimum value.

### HTML

```
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
<input id="rating" type="text" />
</td>
</tr>
</table>
</div>
```

### JAVASCRIPT

```
// Add the following script to render Rating with minimum value.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RatingComponent {
$(function () {
var sample = new ej.Rating($("#rating"), {
minValue: 3
});
});
}
```

The following screenshot illustrates **Rating** with **minimum value**.



#### Max Value

**EJ Rating control** provides support for setting **maximum value**. This is achieved by adding the **maxValue** property. By default, **maxValue** is five.

The following code example is used to render the Rating control with **maximum rating**.

Add the following HTML to render Rating with maximum value.

#### HTML

```
<div id="container" style="border: 1px solid black; width: 300px; padding: 2px">
  <table>
    <tr>
      <td valign="top">Rating:
    </td>
    <td>
      <input id="rating" type="text" />
    </td>
    </tr>
  </table>
</div>
```

#### JAVASCRIPT

```
// Add the following script to render Rating with minimum value.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RatingComponent {
  $(function () {
    var sample = new ej.Rating($("#rating"), {
      maxValue: 10
    });
  });
}
```

The following screenshot illustrates the **Rating** with **maximum value**.



### Set Precision

In a real-time movie **Rating** scenario, you can set **Precision** between two whole numbers such as 2.5 or 3.7 and it is done using the property **Precision** by changing the value to **ej.Rating.Precision.Half** or **ej.Rating.Precision.Exact**. By default, **Precision** is **Full**.

The following code example is used to render the **Rating** control with **Precision**.

Add the following HTML to render Rating with Precision.

### HTML

```
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Full Precision:
</td>
<td>
<input id="rating" type="text" />
</td>
</tr>
<tr>
<td valign="top">Half Precision:
</td>
<td>
<input id="halfRating" type="text" />
</td>
</tr>
<tr>
<td valign="top">Exact Precision:
</td>
<td>
<input id="exactRating" type="text" />
</td>
</tr>
</table>
</div>
```

### JAVASCRIPT

```
// Add the following script to render Rating with Precision.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RatingComponent {
$(function () {
var sample = new ej.Rating($("#rating"), {
```

```

value: 4
});
var sample1 = new ej.Rating($("#halfRating"), {
precision: ej.Rating.Precision.Half,
value: 3.5
});
var sample2 = new ej.Rating($("#exactRating"), {
precision: ej.Rating.Precision.Exact,
value: 3.7
});
});
}

```

The following screenshot illustrates the **Rating with Precision**.



#### Increment Step

**EJ Rating** control supports customized **increment** value for Rating. This is achieved by adding the **incrementStep** property.

The following code example is used to render the **Rating** control with customized **increment**.

Add the following HTML to render Rating with customized increment.

#### HTML

```

<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
<input id="rating" type="text" />
</td>
</tr>
</table>
</div>

```

#### JAVASCRIPT



```
// Add the following script to render Rating with customized increment.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RatingComponent {
  $(function () {
    var sample = new ej.Rating($("#rating"), {
      incrementStep: 2,
      maxValue: 10
    });
  });
}
```

The following screenshot illustrates the **Rating** with customized increment.



#### Resetting values

**EJ Rating** control provides support for value **reset** at runtime. This is achieved by enabling the **allowReset** property to be **'True'**. By default, the property value is set to **'True'**.

The following code example is used to render the **Rating** control with **allowReset**.

Add the following HTML to render Rating with allowReset.

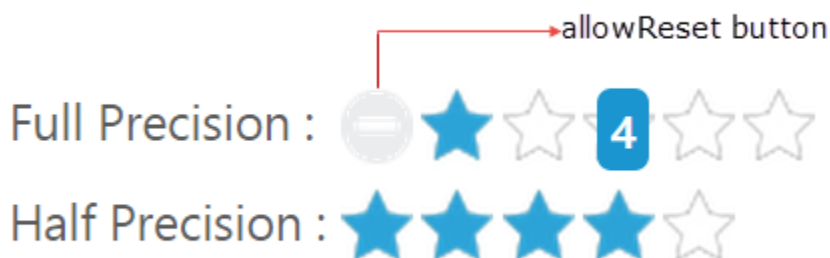
#### JAVASCRIPT

```
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
  <table>
    <tr>
      <td valign="top">Rating:
    </td>
    <td>
      <input id="rating" type="text" />
    </td>
    </tr>
    <tr>
      <td valign="top">Rating:
    </td>
    <td>
      <input id="rest" type="text" />
    </td>
    </tr>
  </table>
</div>
```

#### JAVASCRIPT

```
// Add the following script to render Rating with allowReset.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RatingComponent {
  $(function () {
    var sample = new ej.Rating($("#rating"), {
      allowReset: true
    });
    var sample = new ej.Rating($("#rest"), {
      allowReset: false
    });
  });
}
```

The following screenshot illustrates the **Rating** with **allowReset**.



#### Read only

**Rating** control provides support for changeable or unchangeable values for **Rating** control. This is achieved by the **readOnly** property. When this property is set to **'True'** the **Rating** value becomes unchangeable. By default, this property value is set to **'False'**.

The following code example is used to render the **Rating** control with **readOnly**.

Add the following HTML to render Rating with **readOnly**.

#### HTML

```
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
  <table>
  <tr>
  <td valign="top">Rating:
  </td>
  <td>
  <input id="rating" type="text" />
  </td>
  </tr>
  </table>
</div>
```

**JAVASCRIPT**

```
// Add the following script to render Rating with readOnly .
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RatingComponent {
$(function () {
var sample = new ej.Rating($("#rating"), {
readOnly: true
});
});
}
```

The following screenshot illustrates the **Rating** with **readOnly**.



**Enable or Disable**

**Rating** control provides support to **enable** or **disable** the control. This is achieved by the **enabled** property. By default the property value is **'True'**.

The following code example is used to render the **Rating** control with **enable** or **disable** support.

Add the following HTML to render Rating with enable or disable support.

**HTML**

```
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
<input id="rating" type="text" />
</td>
</tr>
</table>
</div>
```

**JAVASCRIPT**

```
// Add the following script to render Rating in disabled form.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RatingComponent {
$(function () {
var sample = new ej.Rating($("#rating"), {
enabled: false
});
});
}
```

The following screenshot illustrates the **Rating** in **disabled** form.



## Appearance and Styling

### Show ToolTip

**Rating** control provides support for **Tooltip** values. This is achieved by enabling the **showTooltip** property to be **'True'**. When you move the mouse over the Rating control it displays the Tooltip value as rating value. By default this property value is set to **'True'**.

The following code example is used to render the **Rating** control without **tooltip**.

Add the following HTML to render Rating with Tooltip.

### HTML

```
<div id="container" style="border: 1px solid black; width: 300px; padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
<input id="rating" type="text" />
</td>
</tr>
</table>
</div>
```

Add the following script to render Rating without Tooltip.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RatingComponent {
$(function () {
var sample = new ej.Rating($("#rating"), {
showTooltip:false
});
});
}
```

The following screenshot illustrates **Rating** without **Tooltip**.



### Adjusting Rating Size

#### *Adjust Shape Width and Shape Height*

You can customize the width and height of the **Rating** by **shapeWidth** and **shapeHeight** properties. These properties completely depend on rating image's size. The shapeWidth and shapeHeight are adjusted within the rating image size.

The following code example is used to render the **Rating** control with customized **shapeWidth** and **shapeHeight**.

Add the following HTML to render Rating with customized shapeWidth and shapeHeight.

#### **HTML**

```
<div style="margin-top: 0px;">
<h4>Rating:</h4>
<input id="rating" type="text" class="rating" />
</div>
```

#### **JAVASCRIPT**

```
// Add the following script to render Rating with customized shapeWidth
and shapeHeight.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RatingComponent {
$(function () {
var sample = new ej.Rating($("#rating"), {
value: 4,
shapeWidth: 29,
shapeHeight: 29
});
});
}
```

Add the following styles.

#### **CSS**

```
<style type="text/css">
.e-rating
{
margin-top: -7px;
}
.e-rating.e-horizontal .e-shape-list, .e-rating.e-vertical .e-shape-list,
.e-rating.e-horizontal .e-shape, .e-rating.e-vertical .e-shape, .e-
rating.e-horizontal .e-ul, .e-rating.e-vertical .e-ul, .e-rating.e-
horizontal .e-reset, .e-rating.e-vertical .e-reset
{
```

```

height:28px;width:28px;
background:url(images/crystal-stars.png) no-repeat;
}
.e-rating.e-horizontal .e-reset, .e-rating.e-vertical .e-reset {
background-position: 0 42px;
margin-left: 2px;
}
.e-rating.e-horizontal .e-shape-list
{
background-position: 0 -56px;
}
.e-rating.e-horizontal .e-reset:hover
{
background-position: 0 42px;
}
.e-rating .e-shape.inactive
{
background-position: 0 -56px;
}
.e-rating .e-shape.active {
background-position: 0 -112px;
}
.e-rating .e-shape.selected {
background-position: 0 -84px;
}
.e-tooltip {
background-color:white;
border:2px solid #b0c4de;
color:black
}
</style>

```

The following screenshot illustrates **Rating** with customized **shapeWidth** and **shapeHeight**.

**Rating:**



#### Theme

**Rating** control's style and appearance are controlled based on **CSS** classes. In order to apply styles to the Rating control, refer 2 files namely, **ej.widgets.core.min.css** and **ej.theme.min.css**. When the file **ej.widgets.all.min.css** is referred, then it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.widgets.all.min.css** is the combination of these both.

By default, there are 16 themes support available for **Rating** control namely:

- default-theme
- flat-azure-dark
- fat-lime

- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap
- high-contrast-01
- high-contrast-02
- material
- office-365

### Custom styles

The style of the **Rating** control is customized by **cssClass** property.

The following code example is used to render the **Rating** control with **customized style**.

Add the following HTML to render Rating with customized style.

#### HTML

```
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
<input id="rating" type="text" />
</td>
</tr>
</table>
</div>
```

#### JAVASCRIPT

```
// Add the following script to render Rating with customized style.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RatingComponent {
  $(function () {
    var sample = new ej.Rating($("#rating"), {
      cssClass: "custom"
    });
  });
}
```

Add the following styles.

#### CSS

```
<style type="text/css">
.custom {
background-color: greenyellow;
}
</style>
```

The following screenshot illustrates the **Rating** with customized style.



## Orientation

**Rating** provides support for **vertical** orientation. By default Rating renders with **horizontal** orientation. You can change the orientation by the **orientation** property.

The following code example is used to render the Rating control with **vertical orientation**.

Add the following HTML to render Rating with customized orientation.

### HTML

```
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
<input id="rating" type="text" />
</td>
</tr>
</table>
</div>
```

### JAVASCRIPT

```
// Add the following script to render Rating with customized orientation.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RatingComponent {
$(function () {
var sample = new ej.Rating($("#rating"), {
orientation: "vertical"
});
});
}
```

The following screenshot illustrates the **Rating** with **vertical orientation**.





## ReportViewer

### Overview

The ReportViewer is a visualization control to view Microsoft SSRS RDL/RDLC files on a web page and it is powered by HTML5/JavaScript. It has support to bind DataSources/Parameters to the Reports and also supports exporting, paging, zooming and printing the report.

### Key Features

The ReportViewer supports the following features.

- DataSources (SQL Server, Oracle, MS Azure, XML, Business Object and SQL Server Compact Data Sources)
- Filters and Parameters
- Built-in Fields and Expressions
- Grouping and Sorting
- Report Items
- Table
- Matrix
- List
- Chart
- Sparkline
- DataBar
- Map
- Gauge
- Indicator
- Image
- Textbox
- Line
- Rectangle
- Subreport
- Actions: Drilldown, Drill through, Toggle and Document Map
- Printing, Exporting, Paging, FitToPage and Zooming
- Report Processing Events
- KnockOut/AngularJS Support

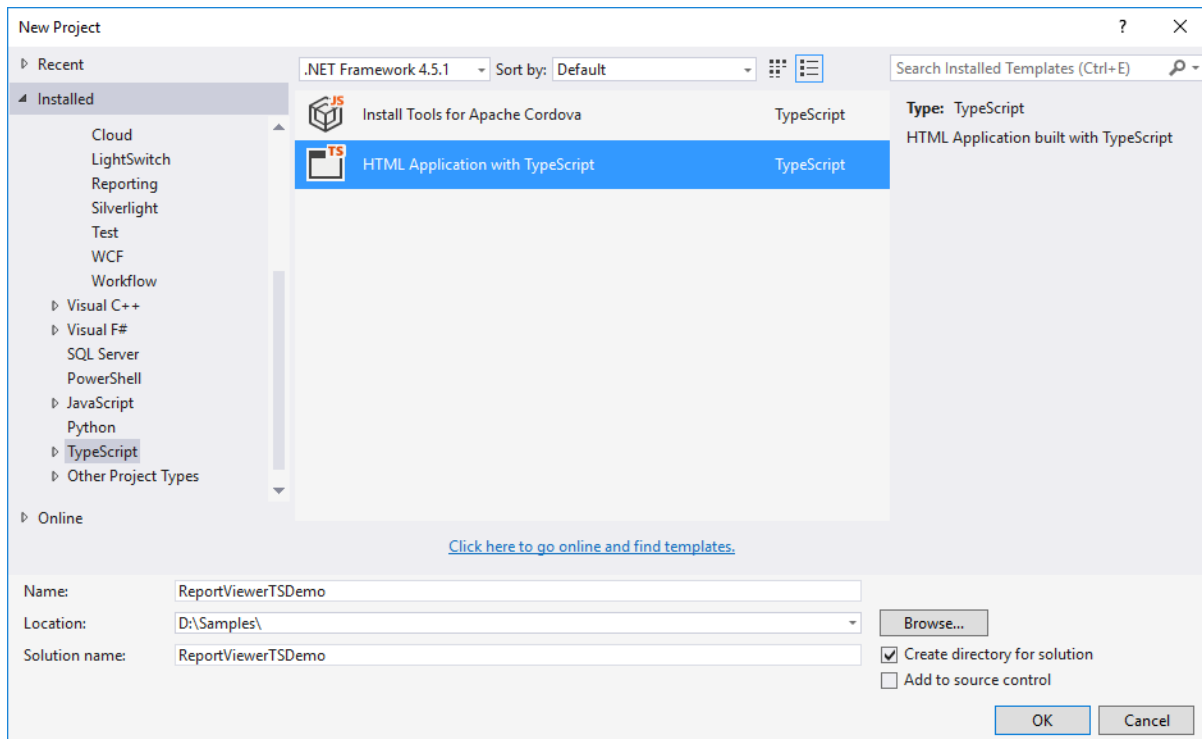
- Toolbar Customization

## Getting Started

This section explains briefly about how to create a ReportViewer in your web application Typescript.

### Create your first ReportViewer in TypeScript

Open Visual Studio and create a new project by clicking New Project. Select the TypeScript category, choose the HTML Application with TypeScript template, and then click OK. The following screenshot displays the Project Creation Wizard.



### Project Creation Wizard

For common steps of typescript , you can also refer [here](#).

The default type definition file `ej.web.all.d.ts` needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the `ej.web.all.d.ts` file into your project and then need to refer it in your TypeScript application (`app.ts` file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the `ej.web.all.d.ts` file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from `ej.web.all.d.ts` file, it is also necessary to make use of the `jquery.d.ts` file in your TypeScript application, which can be downloaded from [here](#).

### Script and CSS Reference

Add the scripts and CSS references to the "index.html" page as the order mentioned in the following code example.

**HTML**

```

<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{ { site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
<script
src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.valida
te.min.js"></script>
<script src="http://cdn.syncfusion.com/{ { site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>

```

In the above code, `ej.web.all.min.js` script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

*Control Initialization*

Add a `div` container to render the ReportViewer.

**HTML**

```

<!DOCTYPE html>
<html>
<body>
<div id="groupingAggregate"></div>
</body>
</html>

```

Initialize the ReportViewer in `app.ts` file by using the `ej.ReportViewer` method.

**HTML**

```

/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module ReportViewerComponent {
$(function () {
var report = new ej.ReportViewer($("#groupingAggregate"), {
reportServiceUrl: 'http://js.syncfusion.com/ejservices/api/ReportViewer',
processingMode: ej.ReportViewer.ProcessingMode.Remote,
reportPath: 'GroupingAgg.rdl',
isResponsive: true
});
});
}

```

**Note:** Default RDL Report will be rendered, which is used in the online service. You can obtain sample rdl/rdlc files from Syncfusion installed location  
 (%userprofile%\AppData\Local\Syncfusion\EssentialStudio\{{ site.releaseversion }}\Common\Data\ejReportTemplate).

### Run the Application

Run the sample application and you can see the ReportViewer on the page as displayed in the following screenshot.

| Product Category | Product Sub Category | Quarterly Month | Sales                 |
|------------------|----------------------|-----------------|-----------------------|
| Bikes            | Road Bikes           | Q1              | \$3,171,787.61        |
| Accessories      | Helmets              | Q1              | \$4,945.69            |
| Clothing         | Jerseys              | Q1              | \$9,517.33            |
| <b>Q1 Sales</b>  |                      |                 | <b>\$3,186,250.64</b> |
| Components       | Road Frames          | Q2              | \$155,311.41          |
| Clothing         | Socks                | Q2              | \$1,899.62            |
| Bikes            | Mountain Bikes       | Q2              | \$2,416,836.61        |
| <b>Q2 Sales</b>  |                      |                 | <b>\$2,574,047.64</b> |
| Components       | Forks                | Q3              | \$26,166.78           |
| Clothing         | Tights               | Q3              | \$67,088.30           |
| Accessories      | Locks                | Q3              | \$6,325.00            |
| Components       | Road Frames          | Q3              | \$957,715.19          |
| Components       | Wheels               | Q3              | \$288,627.83          |
| <b>Q3 Sales</b>  |                      |                 | <b>\$1,345,923.11</b> |
| Clothing         | Bib-Shorts           | Q4              | \$35,322.87           |

### ReportViewer with Grouping Aggregate Report

#### Load SSRS Server Reports

ReportViewer supports to load RDL/RDLC files from SSRS Server. The following steps help you to load reports from SSRS Server.

1. Add a `div` container to render the ReportViewer.

#### HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="territoryReportViewer"></div>
</body>
</html>
```

2. Set the `reportPath` from SSRS and `SSRS reportServerUrl` in the ReportViewer properties.

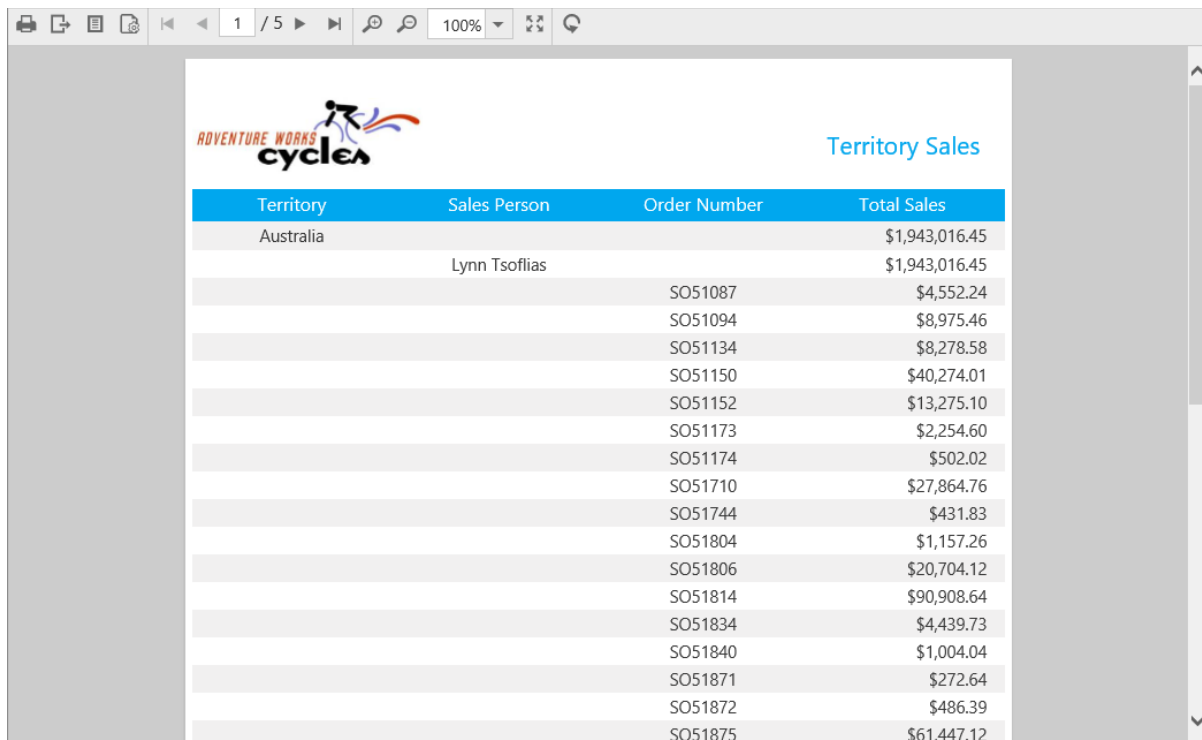
**HTML**

```

/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module ReportViewerComponent {
$(function () {
var report = new ej.ReportViewer($("#territoryReportViewer"), {
reportServiceUrl: 'http://js.syncfusion.com/ejservices/api/ReportViewer',
reportServerUrl: 'http://mvc.syncfusion.com/reportserver',
processingMode: ej.ReportViewer.ProcessingMode.Remote,
reportPath: "/SSRSSamples2/Territory Sales new",
isResponsive: true
});
});
}

```

3. Run the application and you can see the ReportViewer on the page as displayed in the following screenshot.



| Territory | Sales Person   | Order Number | Total Sales    |
|-----------|----------------|--------------|----------------|
| Australia |                |              | \$1,943,016.45 |
|           | Lynn Tsofilias |              | \$1,943,016.45 |
|           |                | SO51087      | \$4,552.24     |
|           |                | SO51094      | \$8,975.46     |
|           |                | SO51134      | \$8,278.58     |
|           |                | SO51150      | \$40,274.01    |
|           |                | SO51152      | \$13,275.10    |
|           |                | SO51173      | \$2,254.60     |
|           |                | SO51174      | \$502.02       |
|           |                | SO51710      | \$27,864.76    |
|           |                | SO51744      | \$431.83       |
|           |                | SO51804      | \$1,157.26     |
|           |                | SO51806      | \$20,704.12    |
|           |                | SO51814      | \$90,908.64    |
|           |                | SO51834      | \$4,439.73     |
|           |                | SO51840      | \$1,004.04     |
|           |                | SO51871      | \$272.64       |
|           |                | SO51872      | \$486.39       |
|           |                | SO51875      | \$61,447.12    |

**Report from SSRS****Load RDLC Reports**

The ReportViewer has data binding support to visualize the RDLC reports. The following code example helps you to bind data to ReportViewer.

1. Add a `div` container to render the ReportViewer.

**HTML**

```
<!DOCTYPE html>
<html>
<body>
<div id="areaCharts"></div>
</body>
</html>
```

2. Assign the RDLC report path to ReportViewer's `reportPath` property and set the data sources to the ReportViewer's `dataSources` property and specify the `processingMode` as local.

## HTML

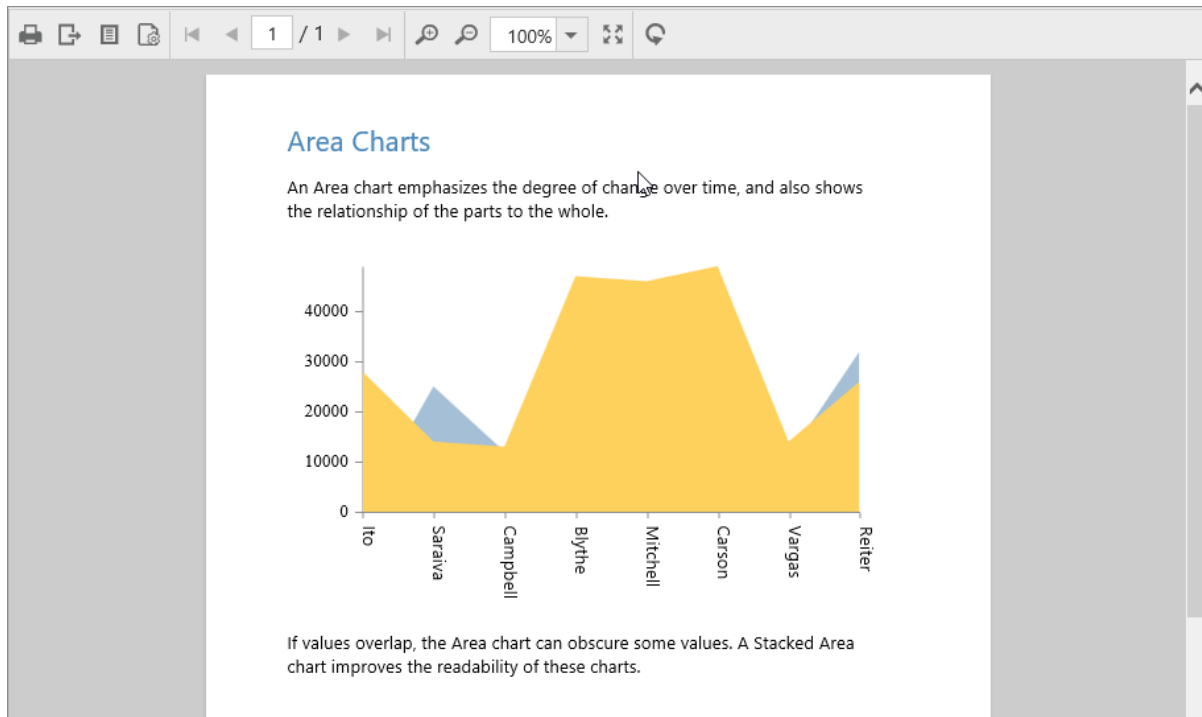
```
/// <reference path="scripts/jquery.d.ts" />
/// <reference path="scripts/ej.widgets.all.d.ts" />
module ReportViewerComponent {
$(function () {
var report = new ej.ReportViewer($("#areaCharts"), {
reportServiceUrl: 'http://js.syncfusion.com/ejservices/api/ReportViewer',
processingMode: ej.ReportViewer.ProcessingMode.Local,
reportPath: 'AreaCharts.rdlc',
dataSource: [{
value: [
{ SalesPersonID: 281, FullName: 'Ito', Title: 'Sales Representative',
SalesTerritory: 'South West', Y2002: 0, Y2003: 28000, Y2004: 3018725 },
{ SalesPersonID: 282, FullName: 'Saraiva', Title: 'Sales Representative',
SalesTerritory: 'Canada', Y2002: 25000, Y2003: 14000, Y2004: 3189356 },
{ SalesPersonID: 283, FullName: 'Cambell', Title: 'Sales Representative',
SalesTerritory: 'North West', Y2002: 12000, Y2003: 13000, Y2004: 1930885
},
{ SalesPersonID: 275, FullName: 'Blythe', Title: 'Sales Representative',
SalesTerritory: 'North East', Y2002: 19000, Y2003: 47000, Y2004: 4557045
},
{ SalesPersonID: 276, FullName: 'Mitchell', Title: 'Sales
Representative', SalesTerritory: 'South West', Y2002: 28000, Y2003:
46000, Y2004: 5240075 },
{ SalesPersonID: 277, FullName: 'Carson', Title: 'Sales Representative',
SalesTerritory: 'Central', Y2002: 33000, Y2003: 49000, Y2004: 3857163 },
{ SalesPersonID: 278, FullName: 'Vargas', Title: 'Sales Representative',
SalesTerritory: 'Canada', Y2002: 11000, Y2003: 14000, Y2004: 1764938 },
{ SalesPersonID: 279, FullName: 'Reiter', Title: 'Sales Representative',
SalesTerritory: 'South East', Y2002: 32000, Y2003: 26000, Y2004: 2811012
}
],
name: 'AdventureWorksXMLDataSet'
}],
isResponsive: true
});
});
}
```

---

**Note:** Default RDLC Report will be rendered, which is used in the online service.

---

3. Run the application and you can see the ReportViewer on the page as displayed in the following screenshot.



### Area Chart RDLC Report

#### Limitations

#### RDL Specification support

- The ReportViewer control does not support RDL Specification for SQL Server 2000 and SQL Server 2005.

#### Layout process

- The Syncfusion ReportViewer control has some limitations in the Tablix cell split layout process in comparison with MS ReportViewer. When the table cell width value exceeds the page width, the entire cell moves to the next page to display the complete cell items.
- The Syncfusion ReportViewer control does not support vertical alignment in the text box report item.

#### Unsupported expression

- The object function and VB function do not have complete support in the ReportViewer.

## How To

### Load unsupported fonts

In RDL, the defined fonts are not supported in cross-platform browsers. The unsupported fonts can be loaded through the **font-face** css rule in the style section of the application.

#### *Using @font-face in Style Section*

- Add the <style> tag to head section of the HTML page.
- Create the CSS rule **font-face** and then add the **font-family** and **src** resources as mentioned below.
- **font-family** -- Specifies a name that will be used as a font face value for font properties.
- **src** -- Specifies the resource containing the font data. This can be a URL to a font file location.
- The following code snippet describes the above steps.

### HTML

```
<style>
@font-face {
font-family: Segoe UI;
src: url(segoue_ui.ttf);
}
</style>
```

## Ribbon

### Overview

The **Ribbon** control for TypeScript provides with rich customizable user interfaces like Office 2010, SharePoint 2010, and Office Web Apps 2010. The Ribbon Tab appears across the top of the page. Each Tab organizes a set of groups that has labels to identify them and also contains a set of controls and group expander.

### Key Features

- **Application Tab** – Represents the menu commands which can used to do any operations, like file related commands. Supports Application Menu and Backstage.
- **Tabs** – Related groups are combined into Tabs for quick and easy access.
- **Contextual Tabs** - Allows to group number of Tabs based on some criteria.
- **Controls' support** - Supports Button, Split button and Dropdown List, Toggle button, and custom controls.
- **Gallery** - Convenient way to visually display related options with text/images.
- **Resize** – Provides resizing to group controls dynamically based on window size.
- **Expand and Collapse** - Ribbon can be expanded and collapsed using expand/collapse button.
- **Screen-Tip** - Supports HTML and enhanced custom tooltips for controls of Ribbon groups.

### Getting Started

For common steps of typescript , you can refer [here](#).

The default type definition file ej.web.all.d.ts needs to include the support for type-checking while initializing any of the Syncfusion widgets.



The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the jquery.d.ts file in your TypeScript application, which can be downloaded from [here](#).

### Script & CSS Reference

Ribbon have the following list of external script dependencies and these should be referred before ej Script files

- [jQuery](#) 1.7.1 and later versions

Also Ribbon have internal dependencies which includes ej.core libraries and [child controls](#). For getting started, you can refer ej.web.all.min.js which includes ej.core and all Syncfusion JavaScript controls.

Add the specific theme reference to your HTML file by referring the appropriate ej.web.all.min.css which contains ej.widgets.core.min.css (layout related CSS) and ej.theme.min.css (theme related CSS) for all the Syncfusion controls.

Create a basic HTML file as shown below to create your Ribbon.

### HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Ribbon - Getting Started</title>
<!-- style sheet for default theme (flat azure) -->
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />
<!--jQuery dependency scripts-->
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js"></script>
<!-- ej script to render JavaScript control-->
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"></script>
</head>
<body>
</body>
</html>
```

**Note:** 1. In case if you don't want to use ej.web.all.min.js file, you can use our [custom script generator](#) to create custom script file with required controls and its dependencies only

2. Ribbon's sample level icons can be loaded using ej.icons.CSS from the location (installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\css\web\ribbon-css"

### Control Initialization

The Ribbon can be configured to the HTML `<div>` element. Add a `<div>` element with Id of `defaultRibbon`.

Ribbon can be initialized with `Application Tab` and UL list is needed for binding menu to application menu which can be specified through `menuItemID` which denotes `id` of UL.

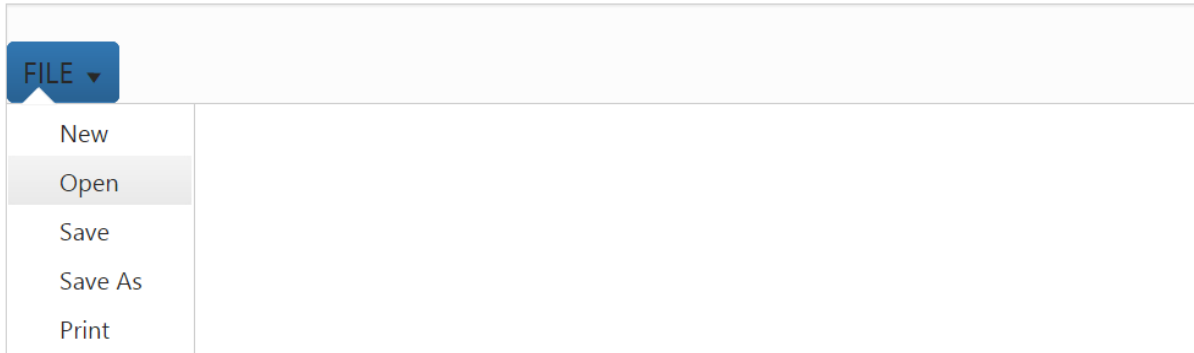
Define the Application Tab with `type` as `menu` to render simple Ribbon control.

### HTML

```
<div>
<div id="defaultRibbon"></div>
</div>
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
<li><a>Save</a></li>
<li><a>Save As</a></li>
<li><a>Print</a></li>
</ul>
</li>
</ul>
<ul id="pasteSplit">
<li><a>Paste</a></li>
</ul>
```

### HTML

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#defaultRibbon"), {
width: "100%",
applicationTab: {
type: ej.Ribbon.ApplicationTabType.Menu, menuItemID: "ribbon"
}
});
});
}
```



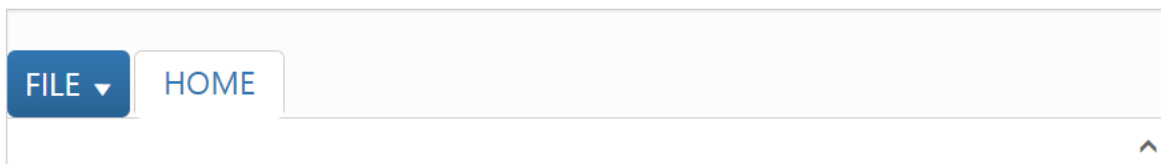
**Note:** Set the required [width](#) to Ribbon, else default parent container or window width will be considered

### Adding Tabs

Tab is a set of related groups which are combined into single item. For creating Tab, [id](#) and [text](#) properties should be specified.

#### HTML

```
module RibbonComponent {
  $(function () {
    var sample = new ej.Ribbon($("#defaultRibbon"), {
      width: "100%",
      applicationTab: {
        type: ej.Ribbon.ApplicationTabType.Menu, menuItemID: "ribbon"
      },
      tabs: [{ id: "home", text: "HOME" }]
    });
  });
}
```



### Configuring Groups

List of controls are combined as logical [groups](#) into Tab. Group alignment type as **row/column**, Default is **row**.

Create group item with [text](#) specified and add content group to Groups collection with ejButton control settings.

#### HTML

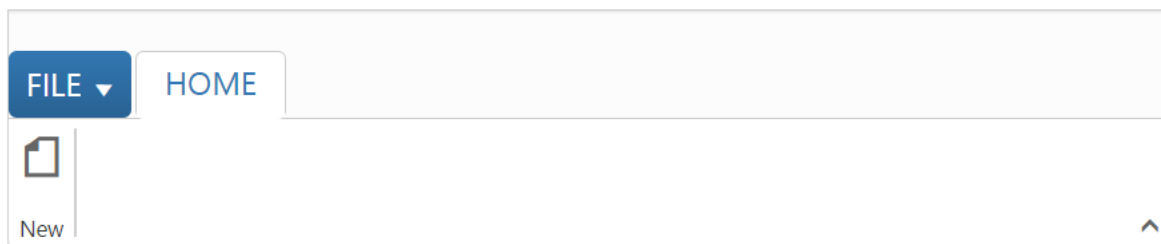
```
module RibbonComponent {
  $(function () {
    var sample = new ej.Ribbon($("#defaultRibbon"), {
      width: "100%",
      applicationTab: {
        type: ej.Ribbon.ApplicationTabType.Menu, menuItemID: "ribbon"
      },

```

```

tabs:
[
  {
    id: "home", text: "HOME",
    groups: [
      {
        text: "New", alignType: ej.Ribbon.AlignType.Rows,
        content: [
          {
            groups: [
              {
                id: "new",
                text: "New",
                buttonSettings: {
                  contentType: ej.ContentType.ImageOnly,
                  prefixIcon: "e-icon e-ribbon e-new"
                }
              }
            ]
          }
        ]
      }
    ]
  }
];

```



### Adding Controls to Group

Syncfusion JavaScript Controls can be added to group's content with corresponding [type](#) specified like button, split button, toggle button, dropdown list, gallery, custom, etc. Default type is **button**.

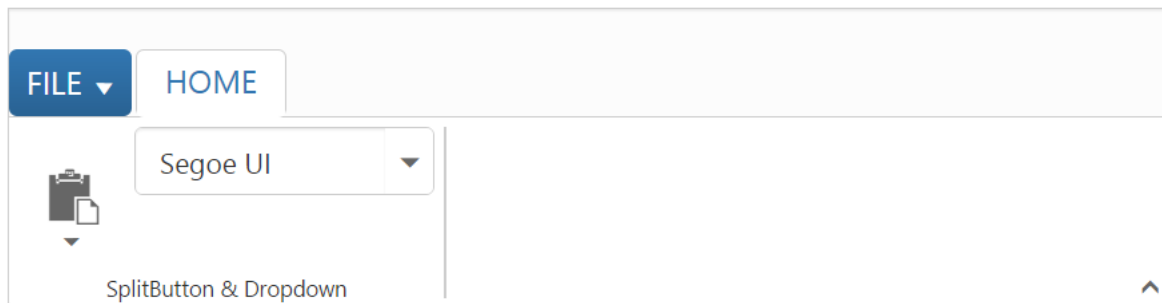
### HTML

```

module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#defaultRibbon"), {
width: "100%",
applicationTab: {
type: ej.Ribbon.ApplicationTabType.Menu, menuItemID: "ribbon"
},
tabs: [
  {
    id: "home", text: "HOME",
    groups: [
      {
        text: "SplitButton & Dropdown", alignType: ej.Ribbon.AlignType.Columns,
        content: [
          {
            groups: [
              {
                id: "paste",
                text: "paste",
                splitButtonSettings: {
                  contentType: ej.ContentType.ImageOnly,
                  prefixIcon: "e-icon e-ribbon e-ribbonpaste",
                  targetID: "pasteSplit",
                  buttonMode: "dropdown",

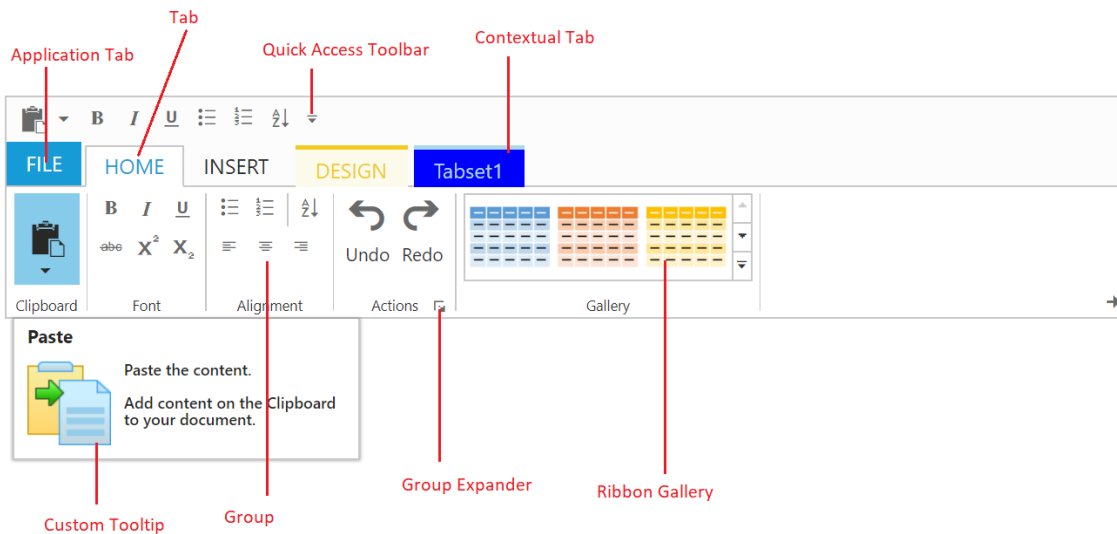
```

```
arrowPosition: ej.ArrowPosition.Bottom,
},
}],
defaults: {
type: ej.Ribbon.Type.SplitButton,
width: 50,
height: 70
},
{
groups: [{
id: "font",
type: ej.Ribbon.Type.DropDownList,
dropdownSettings: {
dataSource: font,
text: "Segoe UI",
width: 150
}
}]
}]
});
});
}
```



### User Interface

Ribbon component able to integrate any custom components and customized their functionality in application end. Our Ribbon component is similar to Microsoft products(Word). The Ribbon UI consists of several sections like Application Tab, Quick Access Toolbar, Tab, Contextual Tab, Gallery and etc.The following screenshot shows the diagrammatic detail of Ribbon UI:



From above screenshot, you can see Ribbon has several subcomponents for different functionalities. The upcoming sections explain the brief details of each functionality and their customizations.

## Ribbon Dependencies

**ej.web.all.js** is a bundle of all Essential JavaScript controls. When you use ej.web.all.js in your application, you can leave this section or else you can try to render ej ribbon in your application by using ej.ribbon file. You can refer to the following frameworks and controls in your project.

| Files                  | Description/Usage  |
|------------------------|--|
| ej.core.min.js         | Must always be referred to before using all the JS controls.   |
| ej.data.min.js         | Used to handle data manager operation and should be used while binding data to JS controls.                          |
| ej.globalize.min.js    | Must be referred to localize any of the JS control's text and content.   |
| ej.ribbon.min.js       | Should be referred when using Ribbon control.  |
| ej.waitingpopup.min.js | This file is used to render waiting popup when on-demand functionality is enabled in ribbon.                         |
| ej.menu.min.js         | This file is used to render menu in the application tab.   |
| ej.scroller.min.js     | This file is used to render scroller in the Ribbon control.  |
| ej.checkbox.min.js     | This file is used to render checkboxes in the Ribbon control.  |
| ej.tab.min.js          | This file is used to render tabs into the Ribbon control.  |
| ej.dropdownlist.min.js | These files are used to render button, split button, toggle button, and dropdown list controls in the ribbon groups. |
| ej.splitbutton.min.js  |  |

|                        |  |
|------------------------|--|
| ej.button.min.js       |  |
| ej.togglebutton.min.js |  |

## Application Tab

The Application Tab is used to represent a **Menu** that do some operations, such as File menu to create, open, and print documents. Application Tab classified by [type](#) property with the following:

- menu
- backstage

## Application Menu

The Application Menu is similar to traditional file menu options and Syncfusion **ejMenu** control is used internally to render this. To show Application Menu in Ribbon, set the [type](#) as **menu** and [menuSettings](#) to customize properties of **ejMenu**.

### \_Create Using Template\_

Set the UL element **id** to [menuItemID](#) property to create Application Menu and it will acts as template to render menu.

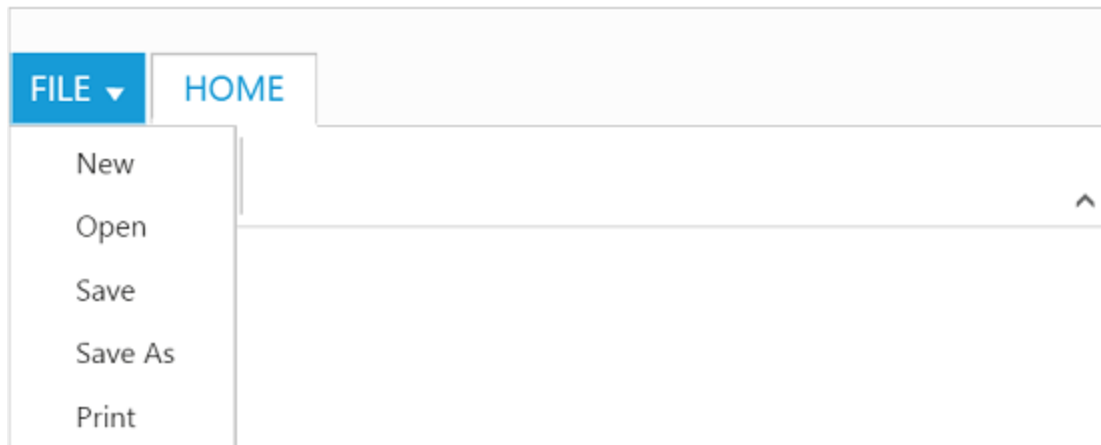
## HTML

```
<div id="Ribbon"></div>
<!--UL template to render menu-->
<ul id="ribbon">
  <li>
    <a>FILE</a>
    <ul>
      <li><a>Open</a></li>
      <li><a>Print</a></li>
    </ul>
  </li>
</ul>
<div id="Contents">Custom control</div>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
  $(function () {
    var sample = new ej.Ribbon($("#Ribbon"), {
      width: "500px",
      applicationTab: {
        type: ej.Ribbon.applicationTabType.menu,
        // menuItemID mapped to UL with id of "menu"
        menuItemID: "ribbon",
        // menu properties defined in menu settings
        menuSettings: {
          openOnClick: false
        }
      },
      tabs: [{
        id: "home",
        text: "HOME",
        groups: [{
```

```

text: "New",
type: "custom",
contentID: "Contents"
}]
}]
});
});
}

```



#### [\\_Binding Data Source\\_](#)

Application Menu can be rendered using JSON Data Source. Please refer [this](#) page to set data source to `ejMenu`.

#### **HTML**

```

<div id="Ribbon"></div>
<ul id="ribbon">
</ul>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
var data = [{
id: 1,
text: "File",
parentId: null
}, {
id: 11,
parentId: 1,
text: "Open"
}, {
id: 12,
parentId: 1,
text: "Save"
}, {
id: 121,
parentId: 12,
text: "Save As"
}, {
id: 122,
parentId: 12,

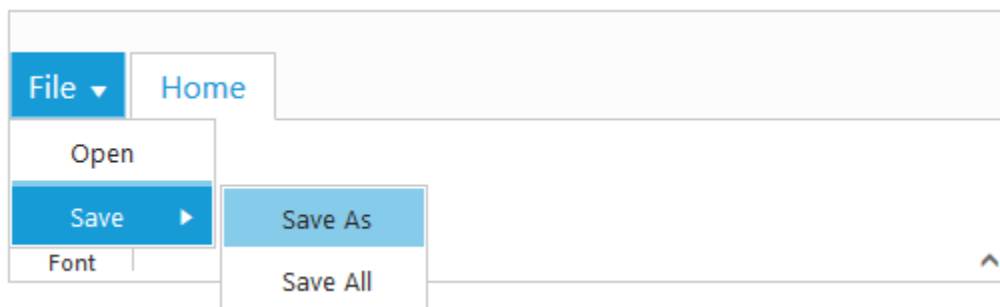
```



```

text: "Save All"
}];
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: 500,
applicationTab: {
type:ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon",
menuSettings: {
// data source to menu
fields: {
dataSource: data,
id: "id",
parentId: "parentId",
text: "text"
}
}
},
tabs: [{
id: "home",
text: "Home",
groups: [{
text: "Font",
content: [{
groups: [{
id: "bold",
text: "Bold",
isBig: true,
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-ribbon e-bold"
}
}
}
}
}
}];
});
});
}

```



## Backstage Page

The Backstage page is where documents and related data of those can be managed, such as Create, Save and other information.

The Backstage page has a feature to add custom Control in left side of the page which contains menu items and the right side contains corresponding user controls.

You can set Application Tab [type](#) as `backstage` and set [id](#) , [text](#) to backstage items. Backstage [pages](#) can be added with required [itemType](#) and [contentID](#) as template id to render template into Backstage.

Separator between Backstage items can be enabled by setting [enableSeparator](#) as true. Width of backstage side header can be customized using [headerWidth](#), If not set based on content given width will be considered.

To render the Ribbon with the Backstage page, refer to the following code snippet.

### HTML

```
<div id="Ribbon"></div>
<div id="newCon">
<table>
<tr>
<td>
<button id="btn1" class="e-bsnewbtnstyle">Blank WorkBook</button></td>
</tr>
</table>
</div>
<div id="accountCon">
<div class="e-userDiv">
<span>User Information</span>
<div>
<div class="e-accuser e-newpageicon"></div>
<div class="e-userCon">
<div>user</div>
<div>any@syncfusion.com</div>
</div>
</div>
</div>
</div>
<a href="#">Sign out</a>
</div>
<div id="ribbonContent">Home control</div>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var button = new ej.Button($("#btn1"), {
size: "large",
height: 200,
width: 205,
contentType: "textandimage",
imagePosition: "imagetop",
prefixIcon: "e-icon e-blank e-infopageicon"
});
var sample = new ej.Ribbon($("#Ribbon"), {
width: "500px",
applicationTab: {
type: ej.Ribbon.applicationTabType.backstage,
```

```

text: "FILE",
// height and width of bask stage is set
backstageSettings: {
text: "FILE",
height: 360,
width: 600,
headerWidth: 125,
pages: [{
// id and text of bask stage item
id: "new",
text: "New",
contentID: "newCon"
}, {
id: "close",
text: "Close",
// enable separator is to set separator between backstage items
enableSeparator: true,
backStageItemType: ej.Ribbon.itemType.button
}, {
id: "account",
text: "Office Account",
contentID: "accountCon"
}]
}
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
text: "New",
type: "custom",
contentID: "ribbonContent"
}]
}]
});
});
}

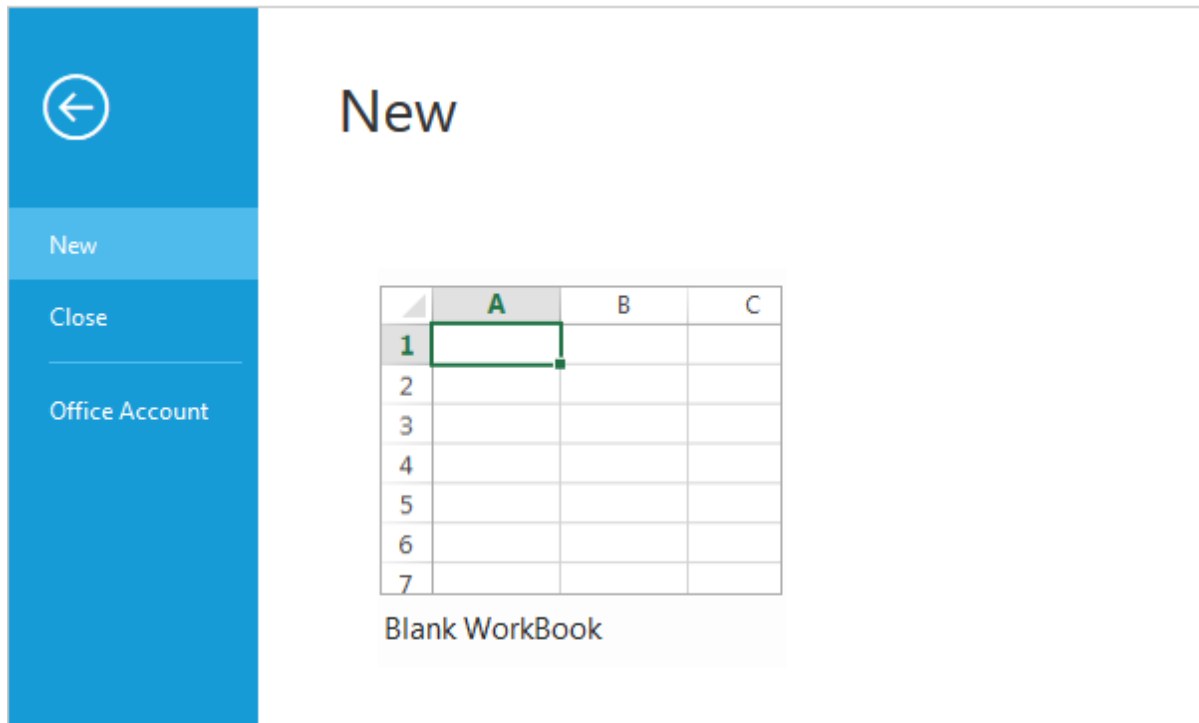
<style type="text/css">
.e-accuser {
background-image: url("../themes/common-images/ribbon/User.jpg");
}
.e-blank {
background-image: url("../themes/common-images/ribbon/blank.png");
}
.e-infopageicon {
background-repeat: no-repeat;
height: 150px;
width: 198px;
}
.e-newpageicon {
background-repeat: no-repeat;
height: 42px;
width: 42px;
}
.e-bspagestyle {
line-height: 0;
font-size: 30px;

```

```

}
.e-ribbon .e-ribbonbackstagepage .e-bsnewbtnstyle {
color: #212121;
background: #fdfdfd;
margin: 20px;
}
</style>

```



**Note:** Height & width of backstage can be set using [height](#) and [width](#), if these are not set, Ribbon's height & width will be considered.

You can add/remove/update backStage item to the ribbon control by using [addBackStageItem](#), [removeBackStageItem](#) and [updateBackStageItem](#) methods. Also you can show/hide the backstage page in ribbon control by using [showBackstage](#) and [hideBackstage](#) (<https://help.syncfusion.com/api/js/ejribbon#methods:hidebackstage> methods).

## Tab

Tab is a collection of control [groups](#) which enables you to organize related commands into single view. Tabs can be added to Ribbon using [tabs](#) property. [id](#) & [text](#) properties are used to set unique ID and header text to Tab.

The manipulation of given text tab in the ribbon control can be done by using [addTab](#), [removeTab](#), [hideTab](#), [showTab](#) methods and [tabAdd](#), [tabCreate](#), [tabRemove](#), [tabClick](#) and [tabSelect](#) events.

## HTML

```

<div id="Ribbon"></div>
<ul id="ribbonMenu">
<li>

```

```
<a>FILE</a>
<ul>
<li><a>Open</a></li>
</ul>
</li>
</ul>
<div id="sendReceive">
Send/Receive All Folders
</div>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: "500px",
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbonMenu"
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
text: "Save",
content: [{
groups: [{
text: "Save",
isBig: true,
buttonSettings: {
prefixIcon: "e-icon e-save",
contentType: ej.ContentType.ImageOnly
}
}]
}]
}, {
text: "Print",
content: [{
groups: [{
text: "Print",
isBig: true,
type: ej.Ribbon.type.toggleButton,
toggleButtonSettings: {
contentType: ej.ContentType.ImageOnly,
defaultText: "Print",
activeText: "Print",
defaultPrefixIcon: "e-icon e-print",
activePrefixIcon: "e-icon e-print",
}
}]
}]
}, {
id: "sendRec",
text: "Send/Receive",
groups: [{
type: "custom",
contentID: "sendReceive"
```

```

}]
}]
});
});
}

```



## Group

[Group](#) is a collection of logical content groups that are combined under related Tab. Each group can be defined using content groups or custom content.

### Adding Tab Groups

Group items can be added to Tabs by specifying [text](#) and corresponding [content](#) to be displayed. The content of group can be specified as either with [content](#) collection, [contentID](#) or [customContent](#). You can add tab group dynamically in the ribbon control with given tab index, tab group object and group index position by using [addTabGroup](#) method.

### Adding Content

Add content to Group item which is based on [type](#) of content specified. The available types are [button](#), [splitButton](#), [toggleButton](#), [gallery](#), and [dropDownList](#).

Groups and defaults settings can be added with the [content](#). You can add group content dynamically in the ribbon control with given tab index, group index, content, content index and sub group index position by using [addTabGroupContent](#).

### \_Defaults\_

The [tabs.groups.content.defaults.height](#), [tabs.groups.content.defaults.width](#),

[tabs.groups.content.defaults.type](#), [tabs.groups.content.defaults.isBig](#) property to the controls in the [group](#) can be specified commonly.

The [height](#) & [width](#) applicable to button, split button, dropdown list ,Toggle button controls and [isBig](#) applicable to only button controls ( button, split , toggle)

### \_Adding Content Groups\_

Controls such as button, split button, dropdown list, toggle button, gallery in the subgroup of the Ribbon tab can be rendered. All of these can be customized using its corresponding settings property such as [buttonSettings](#), [dropdownSettings](#), etc.

Custom controls or items (such as table, div etc.) can be added when the [type](#) set as [custom](#). [defaults](#) control settings of group can be specified for an [individual group](#) instead of specifying them to groups collection commonly.

[Tooltip](#) and [Custom Tooltip](#) can be specified for each group controls.

### HTML

```
<div id="Ribbon"></div>
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>Open</a></li>
</ul>
</li>
</ul>
<ul id="pasteSplit">
<li>Paste
</li>
<li>Paste Special
</li>
</ul>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: "500px",
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon"
},
//tab collection
tabs: [{
id: "home",
text: "HOME",
//group collection of home tab
groups: [{
text: "ClipBoard",
// content collection of group
content: [{
groups: [{
// content group with splitbutton id , text & button settings
id: "paste",
text: "Paste",
toolTip: "Paste",
isBig: true,
type: ej.Ribbon.type.splitButton,
splitButtonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-ribbon e-ribbonpaste",
targetID: "pasteSplit"
}
}]
}]
}, {
text: "Font",
//group align type as columns
alignType: ej.Ribbon.alignType.columns,
content: [{
```

```

groups: [{
  // content group with toggle button settings
  id: "cut",
  toggleButtonSettings: {
    defaultText: "Cut",
    activeText: "Cut Over"
  }
}, {
  id: "copy",
  toggleButtonSettings: {
    defaultText: "Copy",
    activeText: "Copy Over"
  }
}],
// defaults settings of above group's control
defaults: {
  width: 75,
  height: 30,
  type: ej.Ribbon.type.toggleButton
}
}],
}]
});
});
}

```

![(Ribbon/Groupimages/Groupimg1.png)]

#### \_Enable Separator\_

Separates the control from the next control in the group when group `alignType` is `row`. Set “true” to [enableSeparator](#).

#### **HTML**

```

<div id="Ribbon"></div>
<ul id="ribbon">
  <li><a>FILE </a>
  <ul>
    <li><a>New</a></li>
    <li><a>Open</a></li>
  </ul>
</li>
</ul>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
  $(function () {
    var sample = new ej.Ribbon($("#Ribbon"), {
      width: "500",
      applicationTab: {
        type: ej.Ribbon.applicationTabType.menu,
        menuItemID: "ribbon"
      },
      tabs: [{
        id: "home",

```



```

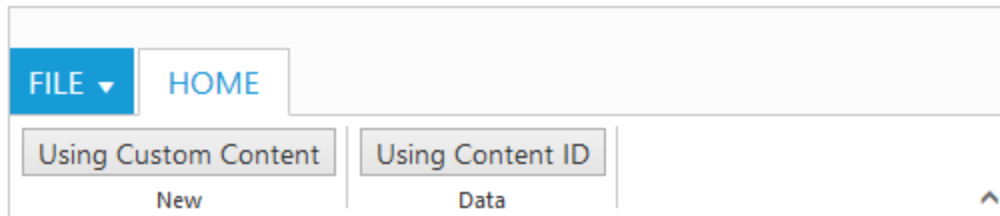
text: "HOME",
groups: [{
text: "New",
// group alignType is "row"
alignType: ej.Ribbon.alignType.rows,
content: [{
groups: [{
id: "new",
text: "New",
toolTip: "New",
// separates New and Font controls
enableSeparator: true,
buttonSettings: {
width: 100,
}
}, {
id: "font",
text: "Font",
toolTip: "Font",
buttonSettings: {
width: 150,
}
}],
defaults: {
type: ej.Ribbon.type.button,
height: 70
}
}]
}]
});
});
}

```



#### Adding Custom Content

Set group [type](#) as `custom` to add custom items such as div, table and custom controls. With type as custom, content can be added in two ways as specified below.



\* HTML contents

can be directly added into the groups as string content using [customContent](#) property \* Custom template id can be specified to render those specific custom template using [contentID](#) property

## HTML

```
<div id="Ribbon"></div>
<ul id="ribbon">
<li><a>FILE </a>
<ul>
<li><a>New</a></li>
</ul>
</li>
</ul>
<button id="button">Using Content ID</button>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: "500",
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon"
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
text: "New",
type: "custom",
customContent: "<button id='customContent'>Using Custom Content</button>"
}, {
text: "Data",
type: "custom",
contentID: "button"
}]
}]
});
});
}
```



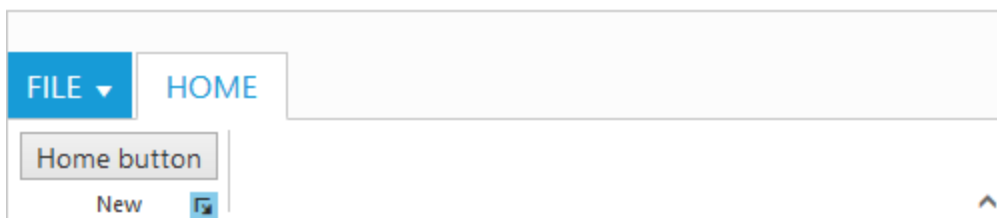
## Group Expander

Set [enableGroupExpander](#) as true to show Group Expander for each group in Tab. These expanders can be customized using [groupExpand](#) event, such as to show popup dialog. To specify tooltip for the group expander of the group [tabs.groups.groupExpanderSettings](#) and

[tabs.groups.groupExpanderSettings.toolTip](#) can be used.

### HTML

```
<div id="Ribbon"></div>
<ul id="ribbon">
<li><a>FILE </a>
<ul>
<li><a>New</a></li>
</ul>
</li>
</ul>
<button id="button">Home button</button>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: "500",
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon"
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
text: "New",
alignType: ej.Ribbon.alignType.rows,
type: "custom",
// group expander enabled
enableGroupExpander: true,
contentID: "button"
}]
}],
// event of group expander
groupExpand: function (args) {
alert("Group expander click triggered");
}
});
});
}
```



### Controls Support

Button, Split Button, DropDownList, Toggle button, Gallery and Custom controls can be added to each groups. You can set [type](#) property in group to define the controls. Default type is button.

### Built in Controls

The following table describes about the built in controls [type](#) and their corresponding control settings.

| Type         | Control Settings  | Example  |
|--------------|---|--|
| Button       | <a href="#">ejButton</a> - <a href="#">buttonSettings</a>             | buttonSettings: { width: 70, contentType: ej.ContentType.ImageOnly, prefixIcon: "e-icon e-ribbon e-new" }  |
| SplitButton  | <a href="#">ejSplitButton</a> - <a href="#">splitButtonSettings</a>   | splitButtonSettings: { contentType: ej.ContentType.ImageOnly, targetID: "pasteSplit", buttonMode: "dropdown", arrowPosition: ej.ArrowPosition.Bottom } |
| ToggleButton | <a href="#">ejToggleButton</a> - <a href="#">toggleButtonSettings</a> | toggleButtonSettings: { contentType: ej.ContentType.ImageOnly, defaultText: "Italic", activeText: "Italic", }  |
| DropDownList | <a href="#">ejDropDownList</a> - <a href="#">dropdownSettings</a>     | dropdownSettings: { dataSource: size, text: "1pt", width: 65 }   |

**Note:** 1. You can specify type either to [group's collection](#) or to each [group](#).

2. For [type](#) property you can assign either string value ("splitbutton") or enum value (ej.Ribbon.type.splitButton).

### HTML

```

<div id="Ribbon"></div>
<ul id="ribbon">
<li><a>FILE</a>
<ul>
<li><a>New</a></li>
</ul>
</li>
</ul>
<ul id="pasteSplit">
<li><a>Paste</a></li>
</ul>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
var fontFamily = ["Segoe UI", "Arial", "Times New Roman", "Tahoma",
"Helvetica"], fontSize = ["1pt", "2pt", "3pt", "4pt", "5pt"], action1 =
["New", "Clear"], action2 = ["Bold", "Italic", "Underline",
"strikethrough", "superscript", "subscript", "JustifyLeft",
"JustifyCenter", "JustifyRight", "JustifyFull", "Undo", "Redo"];
module RibbonComponent {
$(function () {
width: "100%",
applicationTab: { Type: ej.Ribbon.applicationTabType.menu, menuItemID:
"ribbon", menuSettings: { openOnClick: false } },
tabs: [
{
id: "home",

```

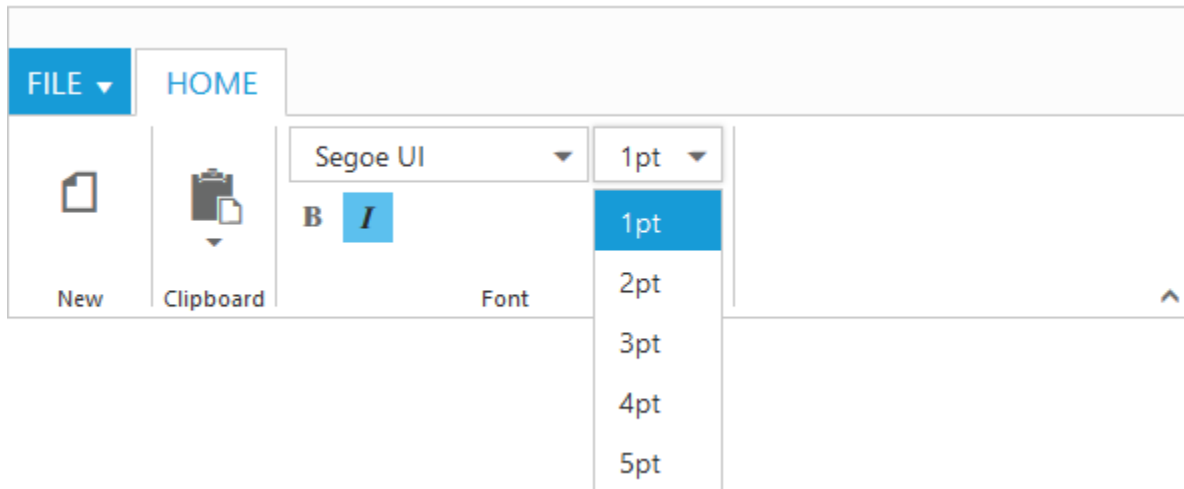
```
text: "HOME",
groups: [
{
text: "New",
alignType: ej.Ribbon.alignType.rows,
content: [
{
groups: [
{
id: "new",
text: "New",
toolTip: "New",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
imagePosition: ej.ImagePosition.ImageTop,
prefixIcon: "e-icon e-ribbon e-new"
}
}
],
defaults: {
type: ej.Ribbon.type.button,
width: 60,
height: 70
}
}
],
},
{
text: "Clipboard",
alignType: ej.Ribbon.alignType.columns,
content: [
{
groups: [
{
id: "paste",
text: "paste",
toolTip: "Paste",
splitButtonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-ribbon e-ribbonpaste",
targetID: "pasteSplit",
buttonMode: "dropdown",
arrowPosition: ej.ArrowPosition.Bottom
}
}
],
defaults: {
type: ej.Ribbon.type.splitButton,
width: 50,
height: 70
}
}
],
},
{
text: "Font",
alignType: "rows",
```

```
content: [
  {
    groups: [
      {
        id: "fontFamily",
        tooltip: "Font",
        dropdownSettings: {
          dataSource: fontFamily,
          text: "Segoe UI",
          width: 150
        }
      },
      {
        id: "fontSize",
        tooltip: "FontSize",
        dropdownSettings: {
          dataSource: fontSize,
          text: "1pt",
          width: 65
        }
      }
    ],
    defaults: {
      type: ej.Ribbon.type.dropDownList,
      height: 28
    }
  }, {
    groups: [
      {
        id: "bold",
        tooltip: "Bold",
        type: ej.Ribbon.type.toggleButton,
        toggleButtonSettings: {
          contentType: ej.ContentType.ImageOnly,
          defaultText: "Bold",
          activeText: "Bold",
          defaultPrefixIcon: "e-icon e-ribbon bold",
          activePrefixIcon: "e-icon e-ribbon bold"
        }
      },
      {
        id: "italic",
        tooltip: "Italic",
        type: ej.Ribbon.type.toggleButton,
        toggleButtonSettings: {
          contentType: ej.ContentType.ImageOnly,
          defaultText: "Italic",
          activeText: "Italic",
          defaultPrefixIcon: "e-icon e-ribbon e-ribbonitalic",
          activePrefixIcon: "e-icon e-ribbon e-ribbonitalic"
        }
      }
    ],
    defaults: {
      isBig: false,
    }
  }
]
```

```

]
}
],
}
],
}
});
}

```



### Custom

You can set [type](#) as `custom` to render custom controls and Custom element id has to be specified as [contentID](#). You can change the element defined in the custom template to appropriate Syncfusion control in the event of Ribbon [create](#).

### HTML

```

<div id="Ribbon"></div>
<ul id="ribbon">
<li>
<a>FILE </a>
<ul>
<li><a>New</a></li>
<li><a>Print</a></li>
</ul>
</li>
</ul>
<input id="fontColor" />
<table id="design" class="e-designstyle">
<tr>
<td>
<input type="checkbox" id="check1" /><label for="check1">Header
Row</label></td>
<td>
<input type="checkbox" id="Check2" checked="checked" /><label
for="Check2">First Column</label></td>
</tr>
<tr>
<td>
<input type="checkbox" id="check4" checked="checked" /><label
for="check4">Total Row</label></td>

```

```

<td>
<input type="checkbox" id="Check5" /><label for="Check5">Last
Column</label></td>
</tr>
</table>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: "600",
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon"
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
text: "Font",
content: [{
groups: [{
id: "fontColor",
toolTip: "Font Color",
contentID: "fontColor"
}],
// defaults settings to controls
defaults: {
height: 30,
type: ej.Ribbon.type.custom
}
}], {
text: "Operators",
content: [{
groups: [{
id: "design",
type: ej.Ribbon.type.custom,
contentID: "design"
}],
}],
}],
// event of Ribbon create to initialize custom control
create: "createControl"
});
});
}
function createControl(args) {
var ribbon = $("#Ribbon").data("ejRibbon");
$("#fontColor").ejColorPicker({ value: "#FFFF00", modelType: "palette",
cssClass: "e-ribbon", toolIcon: "e-fontcoloricon" });
}

```





## Contextual Tabs

[Contextual Tabs](#) are collection of Tabs that extended styling and can be shown based on some criteria. Contextual Tabs can be added like [tabs](#) including groups and content section. You can set [backgroundColor](#) and [borderColor](#) to highlight them as Tab set. Contextual tabs can be added or set dynamically in ribbon control using [addContextualTabs](#) with it's object and index position.

## HTML

```
<div id="Ribbon"></div>
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
</ul>
</li>
</ul>
<div id="Contents">Custom Control</div>
<div id="headings" class="e-headings">
<div>
<p>Ribbon-Heading</p>
<p>No Spacing</p>
</div>
<div>
<p class="e-strong">Ribbon Heading</p>
<p>Strong</p>
</div>
<div>
<p class="e-emphasis">Ribbon Heading</p>
<p>Emphasis</p>
</div>
</div>
<table id="design" class="e-designstyle">
<tr>
<td>
<input type="checkbox" id="Check2" />
<label for="Check2">First Column</label>
</td>
<td>
<input type="checkbox" id="check4" checked="checked" />
<label for="check4">Total Row</label>
</td>
</tr>
</table>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: "500px",
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon"
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
text: "CustomControls",
type: "custom",
contentID: "Contents"
}]
}],
// contextual Tabs collection
contextualTabs: [{
// contextual tab with custom content
backgroundColor: "#FCFBEB",
borderColor: "#F2CC1C",
tabs: [{
id: "Design",
text: "DESIGN",
groups: [{
text: "Table Style Options",
type: "custom",
contentID: "design"
}]
}]
},
// tab set with background & border color
{
backgroundColor: "blue",
borderColor: "lightblue",
tabs: [{
id: "tabset1",
text: "Tabset1",
groups: [{
text: "Tabset1 Style",
type: "custom",
contentID: "headings"
}]
}, {
id: "tabset2",
text: "Tabset2",
groups: [{
text: "TabSet2 Style",
content: [{
// groups with button controls
groups: [{
id: "uppercase",
text: "Upper Case",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-ribbon e-uppercase"
}
```

```

    }, {
      id: "lowercase",
      text: "Lower Case",
      buttonSettings: {
        contentType: ej.ContentType.ImageOnly,
        prefixIcon: "e-icon e-ribbon e-lowercase"
      }
    }],
    defaults: {
      isBig: true
    }
  }
]
});
});
}

```



## Gallery

Galleries are used to display items that can be arranged in a grid-type layout. Items in the gallery can be customized as [button](#)/[menu](#) to display images, text, or both images and text. You can set [type](#) of group as [gallery](#).

## Gallery Items

[Gallery items](#) are collection of the items to be included in the main gallery. You can set [text](#) and [toolTip](#) to gallery item which can also be customized with [buttonSettings](#).

Number of [columns](#) to display in gallery for each row should be specified and the Number of columns in Expanded State ([expandedColumns](#)) can be customized, if not set, `columns` count will be set as default.

**Note:** The `itemHeight` and `itemWidth` for gallery item can be set, if not set default values will be used.

## HTML

```
<div id="Ribbon"></div>
<ul id="ribbon">
<li><a>FILE</a>
<ul>
<li><a>Open</a></li>
</ul>
</li>
</ul>
```

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: "500",
applicationTab: {
type:ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon"
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
type: "gallery",
text: "Gallery",
content: [{
groups: [{
id: "Gallery",
columns: 2,
itemHeight: 54,
itemWidth: 73,
expandedColumns: 3,
type: ej.Ribbon.type.gallery,
galleryItems: [{
text: "GalleryContent1",
toolTip: "GalleryContent1",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-gallerycontent1 e-gbtnimg",
cssClass: "e-gbtnposition"
}
}], {
text: "GalleryContent2",
toolTip: "GalleryContent2",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-gallerycontent2 e-gbtnimg",
cssClass: "e-gbtnposition"
}
}], {
text: "GalleryContent3",
toolTip: "GalleryContent3",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-gallerycontent3 e-gbtnimg",
cssClass: "e-gbtnposition"
}
}], {
text: "GalleryContent4",
toolTip: "GalleryContent4",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-gallerycontent4 e-gbtnimg",
cssClass: "e-gbtnposition"
}
}]
}]
}
}
}

```

```

}]
}]
}]
}]
});
});
}
<style type="text/css">
.e-gallerycontent1 {
background-position: 0 -105px;
}
.e-gallerycontent2 {
background-position: -69px -105px;
}
.e-gallerycontent3 {
background-position: -136px -105px;
}
.e-gallerycontent4 {
background-position: 0 -53px;
}
.e-gbtnposition {
margin-top: 5px;
}
.e-gbtnimg {
background-image: url("../themes/common-images/ribbon/homegallery.png");
background-repeat: no-repeat;
height: 64px;
width: 64px;
}
</style>

```



Ribbon Gallery.



## Gallery at Expanded State

### Custom Gallery Items

[Custom gallery items](#) are the additional items to be displayed at gallery expanded state. You can set [customItemType](#) as `button` or `menu`, Default is `button`.

You can also set [text](#) and [toolTip](#) to custom gallery item which can also be customized with [buttonSettings](#)/[menuSettings](#) based on the [customItemType](#) specified.

### HTML

```
<div id="Ribbon"></div>
<ul id="ribbon">
<li><a>FILE</a> </li>
</ul>
<ul id="customMenu">
<li>
<a>New Quick Step</a>
<ul>
<li><a>Flag and Move</a></li>
</ul>
</li>
</ul>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: "500",
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon"
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
type: "gallery",
text: "Gallery",
content: [{
groups: [{
id: "Gallery",
columns: 2,
itemHeight: 54,
itemWidth: 73,
expandedColumns: 3,
type: ej.Ribbon.type.gallery,
galleryItems: [{
text: "GalleryContent1",
toolTip: "GalleryContent1",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-gallerycontent1 e-gbtnimg",
cssClass: "e-gbtnposition"
}
}, {
text: "GalleryContent2",
```

```

toolTip: "GalleryContent2",
buttonSettings: {
  contentType: ej.ContentType.ImageOnly,
  prefixIcon: "e-icon e-gallerycontent2 e-gbtnimg",
  cssClass: "e-gbtnposition"
}, {
  text: "GalleryContent3",
  toolTip: "GalleryContent3",
  buttonSettings: {
    contentType: ej.ContentType.ImageOnly,
    prefixIcon: "e-icon e-gallerycontent3 e-gbtnimg",
    cssClass: "e-gbtnposition"
  }, {
    text: "GalleryContent4",
    toolTip: "GalleryContent4",
    buttonSettings: {
      contentType: ej.ContentType.ImageOnly,
      prefixIcon: "e-icon e-gallerycontent4 e-gbtnimg",
      cssClass: "e-gbtnposition"
    }
  ]],
  customGalleryItems: [{
    customItemType: ej.Ribbon.customItemType.menu,
    menuId: "customMenu",
    menuSettings: {
      openOnClick: false
    }
  }, {
    text: "Clear Formatting",
    toolTip: "Clear Formatting",
    customItemType: ej.Ribbon.customItemType.button,
    buttonSettings: {
      cssClass: "e-extrabtnstyle"
    }
  }
]
});
});
}
<style type="text/css">
.e-gallerycontent1 {
background-position: 0 -105px;
}
.e-gallerycontent2 {
background-position: -69px -105px;
}
.e-gallerycontent3 {
background-position: -136px -105px;
}
.e-gallerycontent4 {
background-position: 0 -53px;
}

```

```

.e-gbtnposition {
margin-top: 5px;
}
.e-gbtnimg {
background-image: url("../themes/common-images/ribbon/homegallery.png");
background-repeat: no-repeat;
height: 64px;
width: 64px;
}
.e-extracontent .e-extrabtnstyle {
padding-left: 28px;
text-align: left;
}
</style>

```



## Resize

Ribbon control dynamically resizes to display possible number of controls in the optimal layout as the application window size changes.

As the window is narrowed, controls in the Ribbon will be combined as group button with dropdown arrow, in which controls can be expanded with dropdown arrow.

## Tablet Layout

Set [isResponsive](#) as true to enable responsive layout in Ribbon. If client width is above 420px or control content exceeds the page then, the ribbon will render in Tablet mode.

## HTML

```

<div id="Ribbon"></div>
<ul id="ribbonMenu">
<li><a>FILE </a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
</ul>
</li>
</ul>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />

```

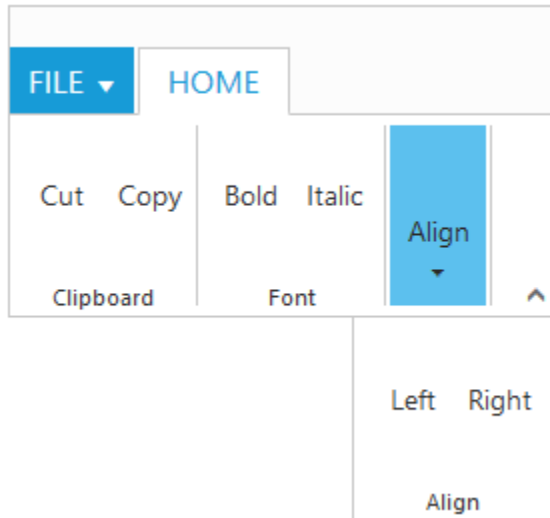


```
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: "20%",
// responsive enabled
isResponsive: true,
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbonMenu",
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
text: "Clipboard",
content: [{
groups: [{
id: "cut",
text: "Cut"
}, {
id: "copy",
text: "Copy"
}],
defaults: {
height: 70,
width: 40
}
}]
}, {
text: "Font",
content: [{
groups: [{
id: "bold",
text: "Bold"
}, {
id: "italic",
text: "Italic"
}],
defaults: {
height: 70,
width: 40
}
}]
}, {
text: "Align",
content: [{
groups: [{
id: "left",
text: " Left"
}, {
id: "right",
text: "Right"
}],
defaults: {
height: 70,
width: 40
}
}]
}]
}
```

```

    }
  }
}
});
});
}

```



### Mobile Layout

If client width is less than 420px, the ribbon will render in mobile mode. In which, you can see that ribbon user interface is customized and redesigned for best view in small screens.

The customized features includes responsive tab & group rendering, backstage, gallery and button controls.

### Responsive Tab and group

Set [isResponsive](#) as true to enable responsive mode in Ribbon.

### HTML

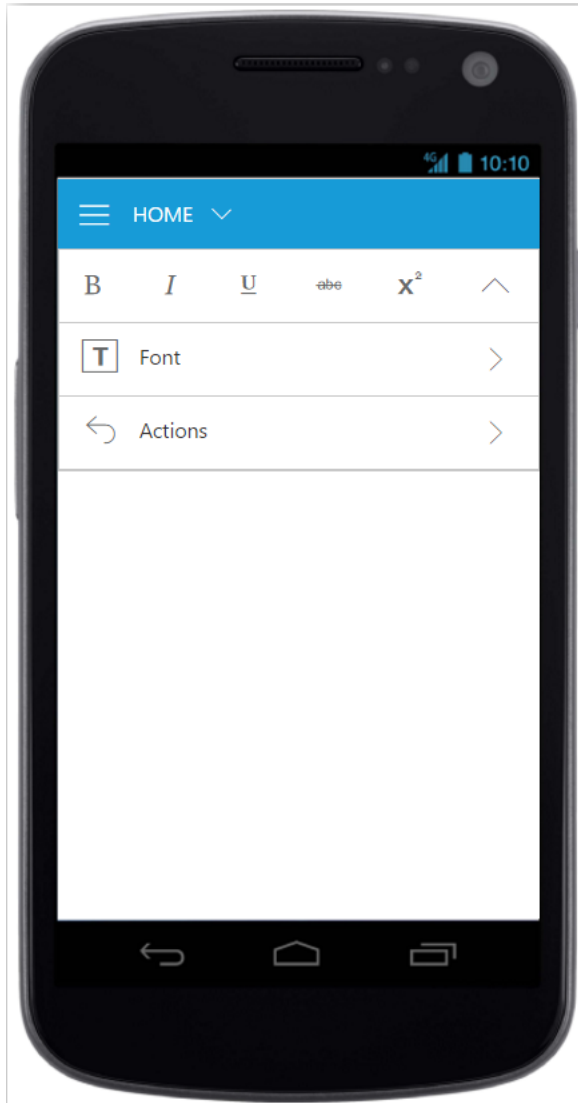
```

<div id="Ribbon"></div>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
  $(function () {
    var sample = new ej.Ribbon($("#Ribbon"), {
      isResponsive:true,
      tabs: [{
        id: "home", text: "HOME", groups: [
          {
            text: "Font", alignType: "rows", content: [{
              groups: [{
                id: "bold",
                type: ej.Ribbon.Type.ToggleButton,
                isMobileOnly: true,
                toggleButtonSettings: {
                  contentType: ej.ContentType.ImageOnly,
                  defaultText: "Bold",
                  activeText: "Bold",

```

```
defaultPrefixIcon: "e-icon e-ribbon e-resbold",
activePrefixIcon: "e-icon e-ribbon e-resbold",
},
{
  id: "italic",
  type: ej.Ribbon.Type.ToggleButton,
  isMobileOnly: true,
  toggleButtonSettings: {
    contentType: ej.ContentType.ImageOnly,
    defaultText: "Italic",
    activeText: "Italic",
    defaultPrefixIcon: "e-icon e-ribbon e-resitalic",
    activePrefixIcon: "e-icon e-ribbon e-resitalic",
    click: "executeAction"
  }
},
{
  id: "underline",
  text: "Underline",
  type: ej.Ribbon.Type.ToggleButton,
  isMobileOnly: true,
  toggleButtonSettings: {
    contentType: ej.ContentType.ImageOnly,
    defaultText: "Underline",
    activeText: "Underline",
    defaultPrefixIcon: "e-icon e-ribbon e-resunderline",
    activePrefixIcon: "e-icon e-ribbon e-resunderline",
  }
},
{
  id: "strikethrough",
  text: "strikethrough",
  isMobileOnly: true,
  type: ej.Ribbon.Type.ToggleButton,
  toggleButtonSettings: {
    contentType: ej.ContentType.ImageOnly,
    defaultText: "Strikethrough",
    activeText: "Strikethrough",
    defaultPrefixIcon: "e-icon e-ribbon strikethrough",
    activePrefixIcon: "e-icon e-ribbon strikethrough",
  }
},
{
  id: "superscript",
  text: "superscript",
  isMobileOnly: true,
  buttonSettings: {
    contentType: ej.ContentType.ImageOnly,
    prefixIcon: "e-icon e-ribbon e-superscripticon",
  }
},
],
defaults: {
  isBig: false
}
}]
```

```
} ,  
]  
}] ,  
});  
});  
}
```



### Ribbon Responsive with tab content

**Note:** To make the Ribbon control to react as responsive in mobile devices, it is necessary to refer the additional `ej.responsive.css` file in the application.

#### Mobile Toolbar Customization

Set `isMobileOnly` as true to group control to show the controls

in the Mobile Toolbar of the ribbon. For each tab, first row of mobile ribbon will pick and display the controls which is set as `isMobileOnly` with look adapt to mobile mode. If `isMobileOnly` property is not

defined to any of the control within tab, then by default first group content will be displayed in first row toolbar.

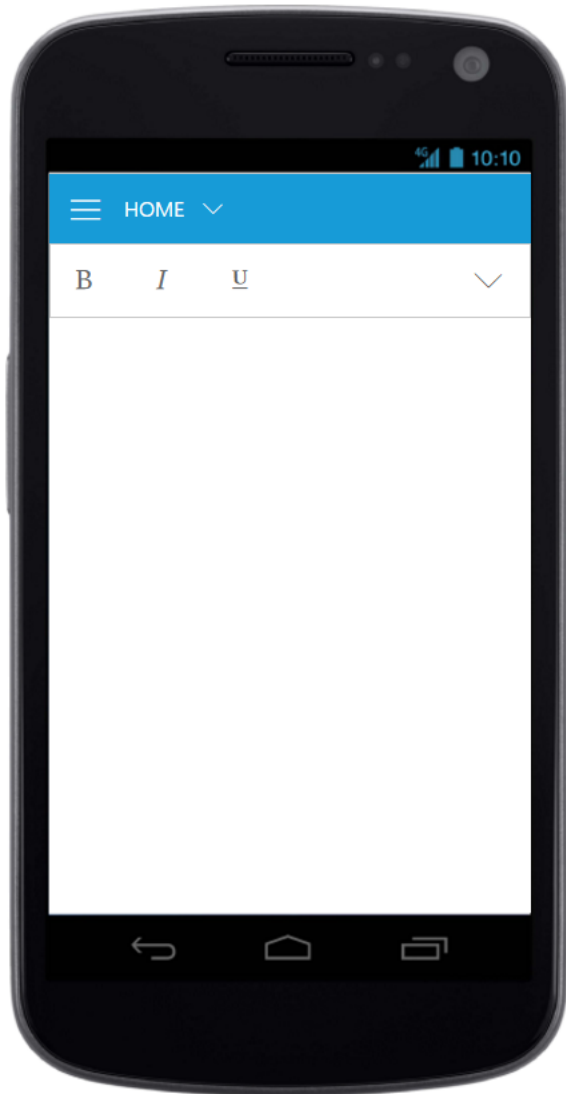
To adapt to proper display of controls , following layout will be customized with constants display.

- First ribbon toolbar and Button controls with min-height.
- Drop down control will adapt to full screen width.
- All button controls icon will be displayed commonly as Top position.

## HTML

```
<div id="Ribbon"></div>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
isResponsive:true,
tabs: [{
id: "home", text: "HOME", groups: [
{
text: "Font", alignType: "rows", content: [{
groups: [{
id: "bold",
type: ej.Ribbon.Type.ToggleButton,
isMobileOnly: true,
toggleButtonSettings: {
contentType: ej.ContentType.ImageOnly,
defaultText: "Bold",
activeText: "Bold",
defaultPrefixIcon: "e-icon e-ribbon e-resbold",
activePrefixIcon: "e-icon e-ribbon e-resbold",
}
},
{
id: "italic",
type: ej.Ribbon.Type.ToggleButton,
isMobileOnly: true,
toggleButtonSettings: {
contentType: ej.ContentType.ImageOnly,
defaultText: "Italic",
activeText: "Italic",
defaultPrefixIcon: "e-icon e-ribbon e-resitalic",
activePrefixIcon: "e-icon e-ribbon e-resitalic",
click: "executeAction"
}
},
{
id: "underline",
text: "Underline",
type: ej.Ribbon.Type.ToggleButton,
isMobileOnly: true,
toggleButtonSettings: {
contentType: ej.ContentType.ImageOnly,
defaultText: "Underline",
activeText: "Underline",
```

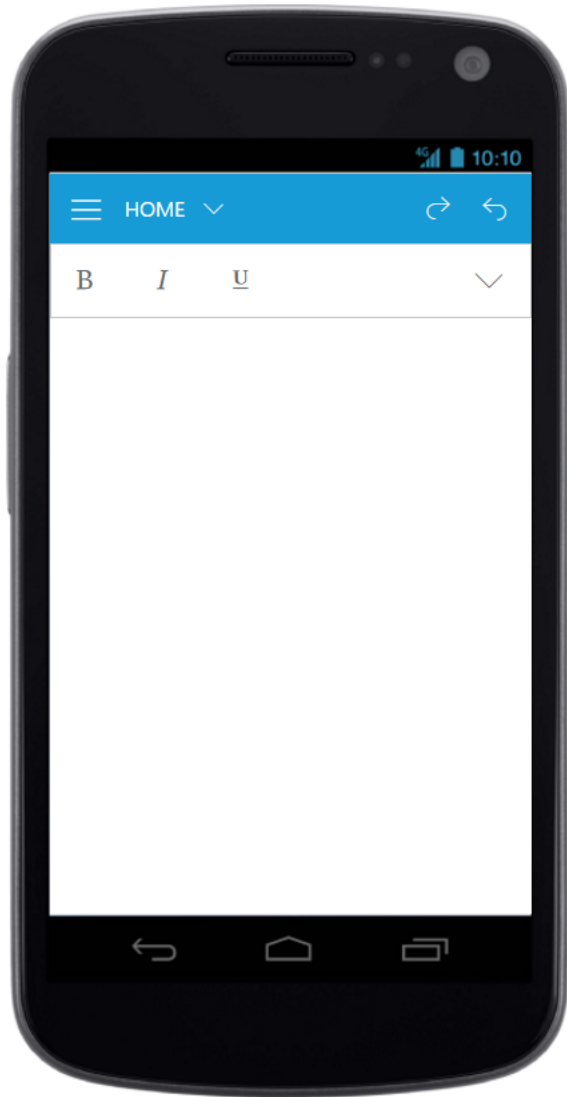
```
defaultPrefixIcon: "e-icon e-ribbon e-resunderline",
activePrefixIcon: "e-icon e-ribbon e-resunderline",
},
{
  id: "strikethrough",
  text: "strikethrough",
  type: ej.Ribbon.Type.ToggleButton,
  toggleButtonSettings: {
    contentType: ej.ContentType.ImageOnly,
    defaultText: "Strikethrough",
    activeText: "Strikethrough",
    defaultPrefixIcon: "e-icon e-ribbon strikethrough",
    activePrefixIcon: "e-icon e-ribbon strikethrough",
  }
},
{
  id: "superscript",
  text: "superscript",
  buttonSettings: {
    contentType: ej.ContentType.ImageOnly,
    prefixIcon: "e-icon e-ribbon e-superscripticon",
  }
},
],
defaults: {
  isBig: false
}
}]
},
]
}],
});
});
}
```



Ribbon Responsive with MobileToolbar

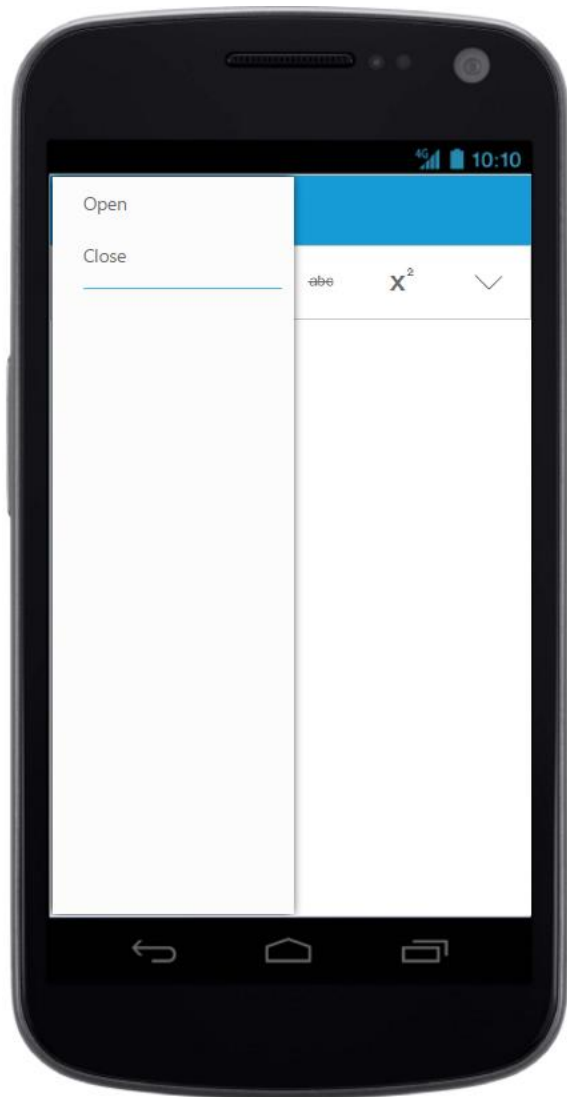
#### *Customized Features*

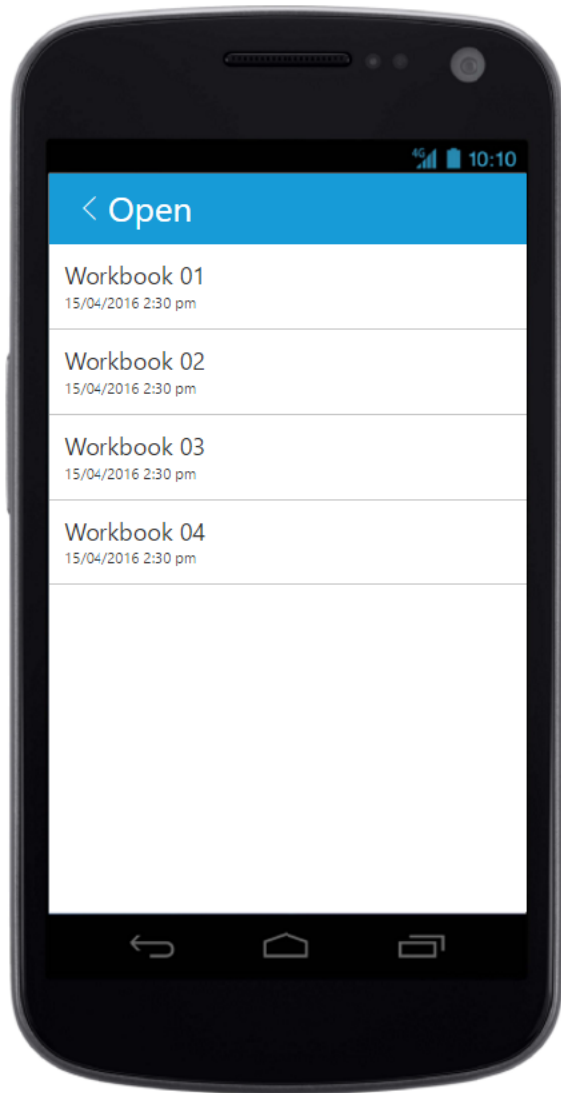
The customized layout for Quick Access Toolbar, backstage, gallery can be seen following screen shots.



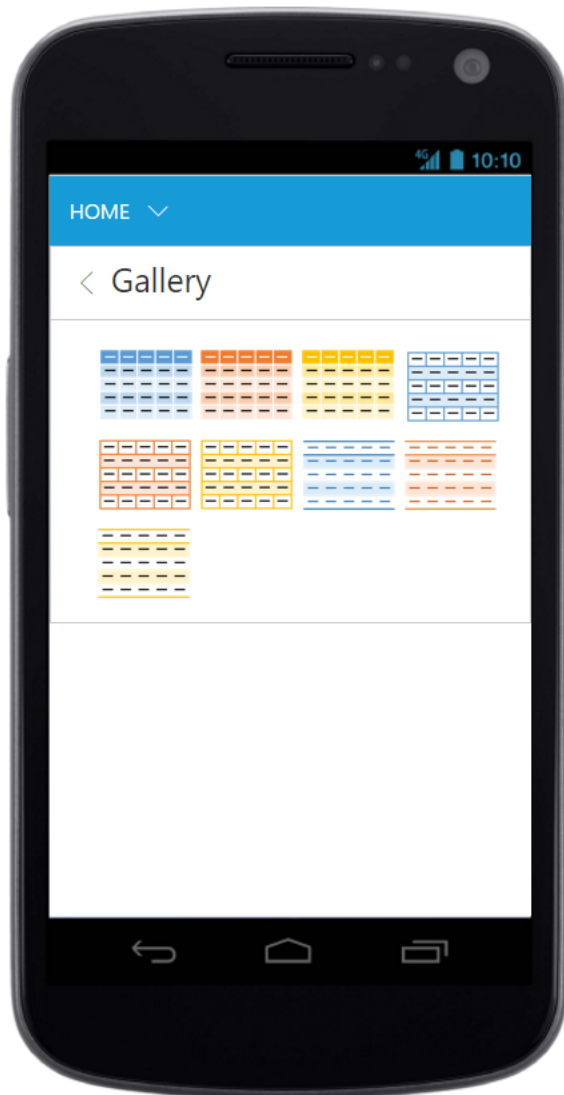
Ribbon Responsive with Quick Access Toolbar







Ribbon Responsive with backstage



### Ribbon Responsive with gallery

#### Group Button Customization

Based on window size, detailed group is shrined into single button and you can expand group items with group button click.

For each group shirked for resizing, Custom Class will be added based on group text. For example, `e-action` whereas `action` is group text. Using this custom class, group button can be customized such as to set icons etc.

#### HTML

```
<div id="Ribbon"></div>
<ul id="ribbonMenu">
  <li>
    <a>FILE</a>
    <ul>
      <li><a>New</a></li>
      <li><a>Open</a></li>
    </ul>
  </li>
</ul>
```

```

</li>
</ul>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
var fontFamily = ["Segoe UI", "Arial"],
    fontSize = ["1pt", "2pt"];
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: "36%",
// resizing enabled
allowResizing: true,
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbonMenu"
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
text: "Clipboard",
alignType: ej.Ribbon.alignType.columns,
enableGroupExpander: true,
content: [{
groups: [{
id: "paste",
text: "paste",
toolTip: "Paste",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-ribbon e-ribbonpaste"
}
}],
defaults: {
isBig: true,
width: 50,
height: 70
}
}, {
groups: [{
id: "cut",
text: "Cut",
toolTip: "Cut",
buttonSettings: {
contentType: ej.ContentType.TextAndImage,
prefixIcon: "e-icon e-ribbon e-ribboncut"
}
}],
}, {
id: "copy",
text: "Copy",
toolTip: "Copy",
buttonSettings: {
contentType: ej.ContentType.TextAndImage,
prefixIcon: "e-icon e-ribbon e-ribboncopy"
}
}],
defaults: {

```

```

width: 60,
height: 40,
isBig: false
}
}]
}, {
text: "Font",
alignType: "rows",
content: [{
groups: [{
id: "fontFamily",
toolTip: "Font",
dropdownSettings: {
dataSource: fontFamily,
text: "Segoe UI",
width: 150
}
}], {
id: "fontSize",
toolTip: "FontSize",
dropdownSettings: {
dataSource: fontSize,
text: "1pt",
width: 65
}
}],
defaults: {
type: ej.Ribbon.type.dropDownList,
height: 28,
isBig: false,
}
}]
}, {
text: "New",
alignType: ej.Ribbon.alignType.rows,
content: [{
groups: [{
id: "new",
text: "New",
toolTip: "New",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
imagePosition: ej.ImagePosition.ImageTop,
prefixIcon: "e-icon e-ribbon e-new"
}
}],
defaults: {
width: 60,
height: 40
}
}]
}, {
text: "Actions",
alignType: ej.Ribbon.alignType.rows,
content: [{
groups: [{
id: "undo",

```

```

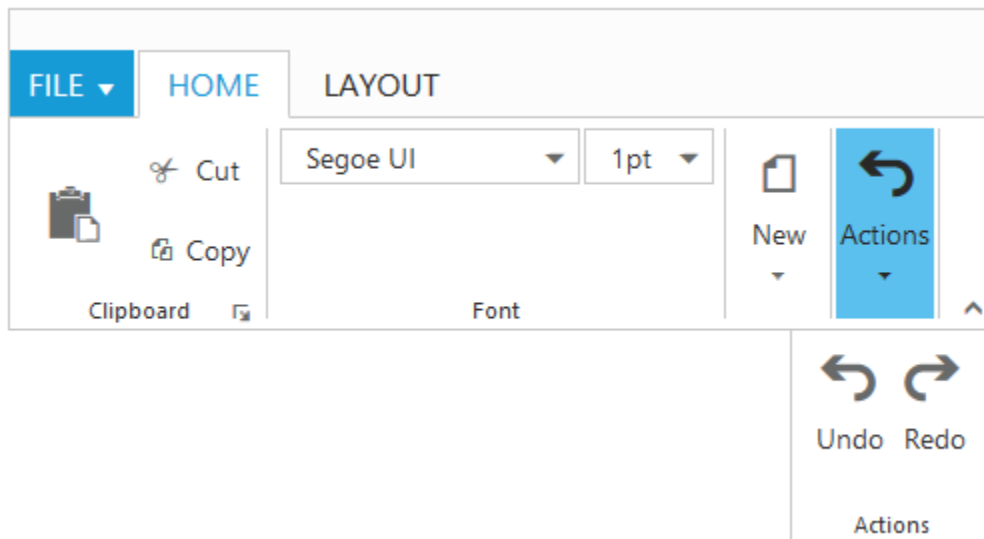
text: "Undo",
toolTip: "Undo",
buttonSettings: {
  contentType: ej.ContentType.TextAndImage,
  imagePosition: ej.ImagePosition.ImageTop,
  prefixIcon: "e-icon e-ribbon e-undo"
}, {
  id: "redo",
  text: "Redo",
  toolTip: "Redo",
  buttonSettings: {
    contentType: ej.ContentType.TextAndImage,
    imagePosition: ej.ImagePosition.ImageTop,
    prefixIcon: "e-icon e-ribbon e-redo"
  }
}],
defaults: {
  width: 40,
  height: 70
}
}]
}], {
  id: "layout",
  text: "LAYOUT",
  groups: [{
    text: "Print Layout",
    alignType: ej.Ribbon.alignType.rows,
    content: [{
      groups: [{
        id: "printLayout",
        text: "Print Layout",
        toolTip: "Print Layout",
        buttonSettings: {
          contentType: ej.ContentType.TextAndImage,
          imagePosition: ej.ImagePosition.ImageTop,
          prefixIcon: "e-icon e-ribbon e-printlayout"
        }
      }
    ]
  }
}],
defaults: {
  width: 80,
  height: 70
}
}]
}
});
});
}
<style type="text/css">
/*styles set to resize group based on group text custom class*/
.e-ribbon .e-New:before, .e-ribbon .e-Actions:before {
font-family: "ej-ribbonfont";
font-size: 28px;
line-height: 35px;
}

```

```

.e-ribbon .e-New:before {
/*e-New to group "New" */
content: "\e167";
text-indent: -1.5px;
}
.e-ribbon .e-Actions:before {
/*e-Actions to group "Actions"*/
content: "\e184";
text-indent: 7px;
}
</style>

```



### Screen Tips

ScreenTip/Tooltip is used to reduce the controls related Help that are needed to the end user to do control related actions.

### HTML Tooltip

Standard `html tooltip` can be set using `tooltip` property of each group item.

### HTML

```

<div id="Ribbon"></div>
<ul id="ribbon">
<li><a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
</ul>
</li>
</ul>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: "20%",
allowResizing: true,

```

```

applicationTab: {
  type: ej.Ribbon.applicationTabType.menu,
  menuItemID: "ribbon",
},
tabs: [{
  id: "home",
  text: "HOME",
  groups: [{
    text: "Clipboard",
    content: [{
      groups: [{
        id: "cut",
        text: "Cut",
        // set tooltip to cut button
        tooltip: "Remove the selection and put it on clipboard"
      },
      {
        id: "copy",
        text: "Copy",
        // set tooltip to copy button
        tooltip: "Put a copy of selection on clipboard",
        buttonSettings: {
          contentType: ej.ContentType.TextAndImage,
          prefixIcon: "e-icon e-ribbon e-ribboncopy"
        }
      }
    ]
  }],
  defaults: {
    height: 70,
    width: 60
  }
}]
}]
});
});
}

```



### Custom Tooltip

Custom Tooltip is used to set detailed help to the user about the controls. You can set [title](#), [content](#) and [prefixIcon](#) class to customize the tooltip with icons.

#### For Groups

[Custom tooltip](#) for each group controls can be specified. Such as to the controls button, split button, dropdown list etc.

### HTML

```

<div id="Ribbon"></div>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
  $(function () {
    var sample = new ej.Ribbon($("#Ribbon"), {
      width: "450",

```



```

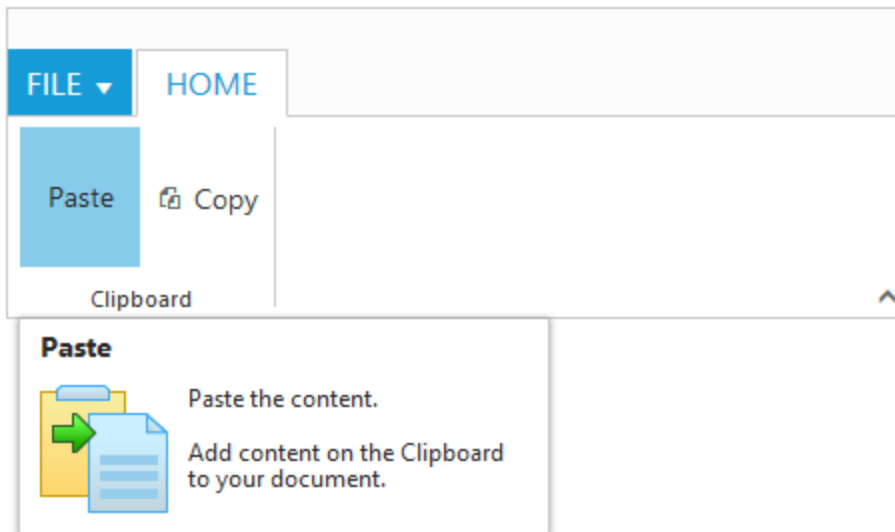
applicationTab: {
  type: ej.Ribbon.applicationTabType.menu,
  menuItemID: "ribbon",
},
tabs: [{
  id: "home",
  text: "HOME",
  groups: [{
    text: "Clipboard",
    content: [{
      groups: [{
        id: "paste",
        text: "Paste",
        // set custom tooltip to paste button with icon settings
        customToolTip: {
          title: "Paste",
          content: "<h6>Paste the content.<br/><br/>Add content on the Clipboard to
your document.</h6>",
          // class to set icon
          prefixIcon: "e-icon e-pastetip"
        }
      }, {
        id: "copy",
        text: "Copy",
        // set custom tooltip to copy button
        customToolTip: {
          title: "Copy",
          content: "<h6>Copy the content.</h6>"
        }
      },
      buttonSettings: {
        contentType: ej.ContentType.TextAndImage,
        prefixIcon: "e-icon e-ribbon e-ribboncopy"
      }
    }
  ]],
  defaults: {
    height: 70,
    width: 60
  }
}]
});
});
<style type="text/css">
.e-pastetip {
background-image: url("../themes/common-images/ribbon/paste.png");
background-repeat: no-repeat;
height: 64px;
width: 64px;
}
</style>
<ul id="ribbon">
<li><a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>

```

```

</ul>
</li>
</ul>

```



*For Gallery*

[Custom tooltip](#) for each [gallery](#) and [custom gallery](#) items button control can be specified.

---

**Note:** Custom gallery item menu is not supported to Custom tooltip.

---

## HTML

```

<div id="Ribbon"></div>
<ul id="ribbon">
<li><a>FILE</a> </li>
</ul>
<ul id="customMenu">
<li>
<a>New Quick Step</a>
<ul>
<li><a>Flag and Move</a></li>
</ul>
</li>
</ul>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: "500",
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon"
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
type: "gallery",

```

```
text: "Gallery",
content: [{
  groups: [{
    id: "Gallery",
    columns: 2,
    itemHeight: 54,
    itemWidth: 73,
    expandedColumns: 3,
    type: ej.Ribbon.type.gallery,
    galleryItems: [{
      text: "Style 1",
      // custom tooltip of style 1 of gallery item
      customToolTip: {
        title: "Style 1",
        content: "<I>Style 1 to customize the table</I>"
      },
      buttonSettings: {
        contentType: ej.ContentType.ImageOnly,
        prefixIcon: "e-icon e-gallerycontent1 e-gbtnimg",
        cssClass: "e-gbtnposition"
      }
    }, {
      text: "Style 2",
      customToolTip: {
        title: "Style 2",
        content: "<I>Style 2 to customize the table</I>"
      },
      buttonSettings: {
        contentType: ej.ContentType.ImageOnly,
        prefixIcon: "e-icon e-gallerycontent2 e-gbtnimg",
        cssClass: "e-gbtnposition"
      }
    }, {
      text: "Style 3",
      customToolTip: {
        title: "Style 3",
        content: "<I>Style 3 to customize the table</I>"
      },
      buttonSettings: {
        contentType: ej.ContentType.ImageOnly,
        prefixIcon: "e-icon e-gallerycontent3 e-gbtnimg",
        cssClass: "e-gbtnposition"
      }
    }, {
      text: "Style 4",
      customToolTip: {
        title: "Style 4",
        content: "<I>Style 4 to customize the table</I>"
      },
      buttonSettings: {
        contentType: ej.ContentType.ImageOnly,
        prefixIcon: "e-icon e-gallerycontent4 e-gbtnimg",
        cssClass: "e-gbtnposition"
      }
    }
  ]],
  customGalleryItems: [{
    text: "Clear Formatting",
```

```

toolTip: "Clear Formatting",
customItemType: ej.Ribbon.customItemType.button,
// custom tooltip of style 1 of custom gallery item
customToolTip: {
  title: "Clear Format",
  content: "<I>To clear formatting</I>"
},
buttonSettings: {
  cssClass: "e-extrabtnstyle"
}, {
  customItemType: ej.Ribbon.customItemType.menu,
  menuId: "customMenu",
  menuSettings: {
    openOnClick: false
  }
}]
}]
}]
}]
});
});
}
<style type="text/css">
.e-gallerycontent1 {
background-position: 0 -105px;
}
.e-gallerycontent2 {
background-position: -69px -105px;
}
.e-gallerycontent3 {
background-position: -136px -105px;
}
.e-gallerycontent4 {
background-position: 0 -53px;
}
.e-gbtnposition {
margin-top: 5px;
}
.e-gbtnimg {
background-image: url("../themes/common-images/ribbon/homegallery.png");
background-repeat: no-repeat;
height: 64px;
width: 64px;
}
.e-extracontent .e-extrabtnstyle {
padding-left: 28px;
text-align: left;
}
</style>

```

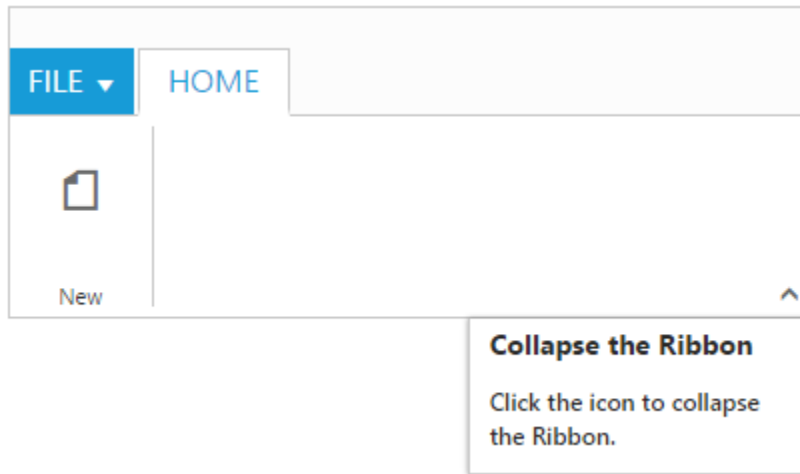


*For Expand Pin*

Specifies the [custom tooltip](#) for expand pin in the Ribbon.



```
});
});
}
```



#### *For Collapse Pin*

Specifies the [custom tooltip](#) for collapse pin in the Ribbon.

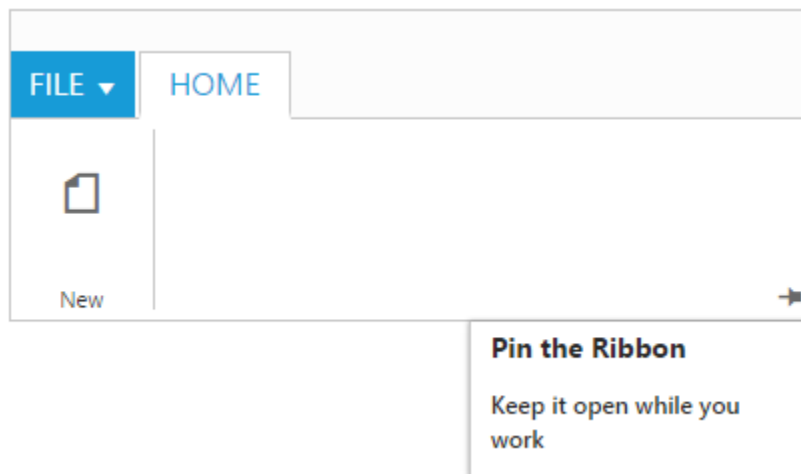
#### **HTML**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
<ul id="ribbon">
<li><a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
</ul>
</li>
</ul>
</div>
<script>
$(function() {
$("#defaultRibbon").ejRibbon({
width: "300",
collapsePinSettings: {
customToolTip: {
title: "Pin the Ribbon",
content: "<h6>Keep it open while you work</h6>"
}
},
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon",
menuSettings: {
openOnClick: false
}
}
},
},
```

```

tabs: [{
  id: "home",
  text: "HOME",
  groups: [{
    text: "New",
    alignType: ej.Ribbon.alignType.rows,
    content: [{
      groups: [{
        id: "new",
        text: "New",
        tooltip: "New",
        buttonSettings: {
          contentType: ej.ContentType.ImageOnly,
          imagePosition: ej.ImagePosition.ImageTop,
          prefixIcon: "e-icon e-ribbon e-new",
          click: "executeAction"
        }
      }],
    }],
    defaults: {
      type: ej.Ribbon.type.button,
      width: 60,
      height: 70
    }
  }],
}];
});

```



*For GroupExpander*

[Custom tooltip](#) for each group expander can be specified.

### HTML

```

<div id="defaultRibbon"></div>
<ul id="ribbon">
<li><a>FILE</a>
<ul>
<li><a>New</a></li>

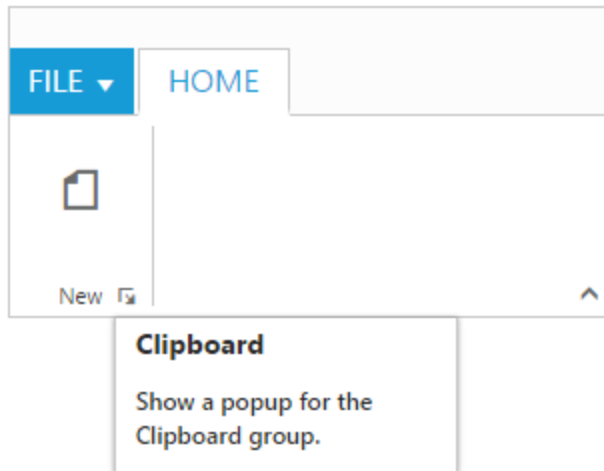
```

```

</li><a>Open</a></li>
</ul>
</li>
</ul>
</div>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#Ribbon"), {
width: "300",
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon",
menuSettings: {
openOnClick: false
}
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
text: "New",
alignType: ej.Ribbon.AlignType.Columns,
enableGroupExpander: true,
groupExpanderSettings: {
customToolTip: {
title: "Clipboard",
content: "<h6>Show a popup for the Clipboard group.</h6>"
}
},
alignType: ej.Ribbon.alignType.rows,
content: [{
groups: [{
id: "new",
text: "New",
toolTip: "New",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
imagePosition: ej.ImagePosition.ImageTop,
prefixIcon: "e-icon e-ribbon e-new",
click: "executeAction"
}
}],
defaults: {
type: ej.Ribbon.type.button,
width: 60,
height: 70
}
}],
}],
});
});
}

```





### Quick Access Toolbar

Quick Access Toolbar provides the shortcuts to the most commonly used commands by placing the controls at the Quick Access Toolbar section. It can be placed at the top or bottom of the Ribbon.

Set [showQAT](#) as true to enable Quick Access Toolbar in Ribbon. It supports the Button, Split Button, Toggle Button controls. The [quickAccessMode](#) is used to change the controls state in Quick Access Toolbar through options as `toolbar`, `menu` and `none`. Default value is `none` and QAT toolbar is created with specified controls added in Toolbar.

The `toolbar` option used to set controls visibility in Quick Access Toolbar. The `menu` option shows the controls in Quick Access Menu and does not show controls in Quick Access Toolbar.

Once the controls are visible in Toolbar, then controls state will be set as ticked in Quick Access Menu and vice versa.

The client side event for Quick Access Toolbar menu click is [gatMenuItemClick](#) and it will be triggered with QAT menu item click.

`More Commands` command provides with Quick Access Menu. This can be customized using [gatMenuItemClick](#) event, such as to show popup dialog.

### HTML

```
<div id="Ribbon"></div>
<ul id="ribbon">
  <li>
    <a>FILE</a>
    <ul>
      <li><a>New</a></li>
    </ul>
  </li>
</ul>
<ul id="split">
  <li><span>Paste</span></li>
</ul>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
  $(function () {
    var sample = new ej.Ribbon($("#Ribbon"), {
```

```

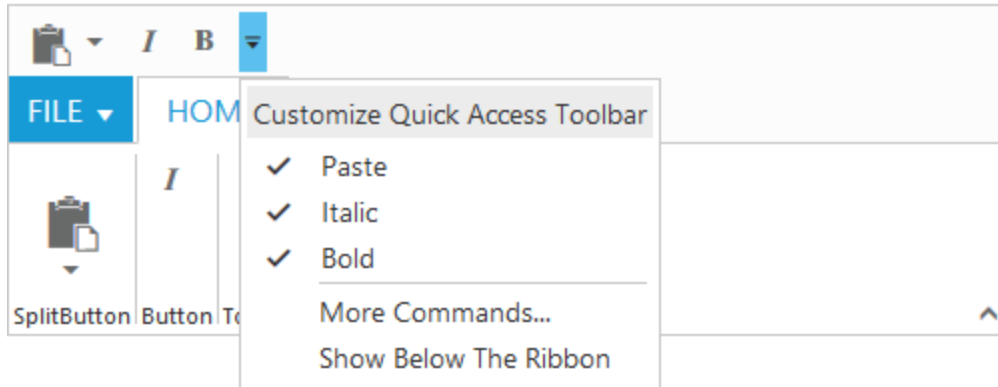
width: 500,
showQAT: true,
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon"
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
text: "SplitButton",
alignType: ej.Ribbon.alignType.columns,
content: [{
groups: [{
id: "paste",
text: "paste",
toolTip: "Paste",
type: ej.Ribbon.type.splitButton,
quickAccessMode: ej.Ribbon.quickAccessMode.toolBar,
splitButtonSettings: {
contentType: ej.ContentType.ImageOnly,
targetID: "split",
prefixIcon: "e-icon e-ribbon e-ribbonpaste",
buttonMode: "dropdown",
arrowPosition: "bottom"
}
}],
defaults: {
type: ej.Ribbon.type.splitButton,
width: 50,
height: 70
}
}],
}, {
text: "Button",
alignType: ej.Ribbon.alignType.rows,
content: [{
groups: [{
id: "italic",
text: "Italic",
toolTip: "Italic",
type: ej.Ribbon.type.button,
quickAccessMode: ej.Ribbon.quickAccessMode.toolBar,
toggleButtonSettings: {
contentType: ej.ContentType.ImageOnly,
defaultText: "Italic",
activeText: "Italic",
prefixIcon: "e-icon e-ribbon e-ribbonitalic"
}
}],
}],
}, {
text: "Toggle",
alignType: ej.Ribbon.alignType.columns,
content: [{
groups: [{
id: "bold",

```

```

toolTip: "Bold",
type: ej.Ribbon.type.toggleButton,
quickAccessMode: ej.Ribbon.quickAccessMode.toolBar,
toggleButtonSettings: {
  contentType: ej.ContentType.ImageOnly,
  defaultText: "Bold",
  activeText: "Bold",
  defaultPrefixIcon: "e-icon e-ribbon bold",
  activePrefixIcon: "e-icon e-ribbon bold"
}
}],
}]
}]
}
});
});
}
<style>
.e-ribbon .e-rbnquickaccessbar .e-ribbonpaste:before {
font-size: 27px;
left: -5px;
top: -6px;
}
.e-ribbon .e-ribbonpaste:before {
font-family: 'ej-ribbonfont';
content: "\e169";
font-size: 36px;
position: relative;
left: -9px;
top: -4px;
}
.e-ribbon .e-ribbonitalic:before ,.e-ribbon .bold:before{
font-family: 'ej-ribbonfont';
font-size: 16px;
left: -1px;
position: relative;
top: -1px;
}
.e-ribbon .e-ribbonitalic:before ,.e-ribbon .bold:before{
content: "\e163";
}
.e-ribbon .bold:before {
content: "\e15a";
}
.e-ribbon .e-rbnquickaccessbar .e-undo::before {
font-size: 18px;
line-height: 12px;
text-indent: -3px;
}
</style>

```



## Globalization and Localization

### Localization

The localization support allows to customize the display of text within the Ribbon in a user-specific culture and locale. The Ribbon control can be localized in specific culture using the common API [locale](#) along with the collection of localized words defined for that culture using the `ej.Ribbon.Locale [culture-code]`. Please find the table with list of properties and its value in locale object.

| Locale key words             | Text                             |
|------------------------------|----------------------------------|
| CustomizeQuickAccess         | Customize Quick Access Toolbar   |
| RemoveFromQuickAccessToolbar | Remove from Quick Access Toolbar |
| AddToQuickAccessToolbar      | Add to Quick Access Toolbar      |
| ShowAboveTheRibbon           | Show Above the Ribbon            |
| ShowBelowTheRibbon           | Show Below the Ribbon            |
| MoreCommands                 | More Commands...                 |

**Note:** By default, the Ribbon control is localized in en-US culture.

For further information on – how to refer the required culture scripts into your application, refer [here](#).

### HTML

```
<div id="defaultRibbon"></div>
<ul id="ribbon">
<li><a>FILE</a>
<ul>
<li><a>New</a></li>
</ul>
</li>
</ul>
</div>
<script>
ej.Ribbon.Locale["es-ES"] = {
CustomizeQuickAccess: "Agordu Rapida Aliro",
RemovefromQuickAccessToolbar: "Forigu de Rapida Aliro Ilobreto",
AddtoQuickAccessToolbar: "Aldoni al Rapida Aliro Ilobreto",
ShowAbovethRibbon: "Montru Super la Ribbona",
```

```
ShowBelowtheRibbon: "Montru Sube la Ribbon",
MoreCommands: "pli Komando"
};
$(function() {
$("#defaultRibbon").ejRibbon({
width: "600",
locale: "es-ES",
showQAT: true,
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon",
menuSettings: {
openOnClick: false
}
},
tabs: [{
id: "home",
text: "HOME",
groups: [{
text: "Clipboard",
alignType: ej.Ribbon.alignType.columns,
content: [{
groups: [{
id: "paste",
text: "paste",
toolTip: "Paste",
quickAccessMode: ej.Ribbon.quickAccessMode.toolBar,
splitButtonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-ribbon e-ribbonpaste",
targetID: "pasteSplit",
buttonMode: "dropdown",
arrowPosition: ej.ArrowPosition.Bottom,
click: "executeAction"
}
}
}],
}
}],
defaults: {
type: ej.Ribbon.type.splitButton,
width: 50,
height: 70
}
}
}
}
});
});
</script>
```



### Right to Left - RTL

By default, Ribbon render its content and layout from left to right. To customize Ribbon direction, you can change direction from LTR to RTL by using [enableRTL](#) as true.

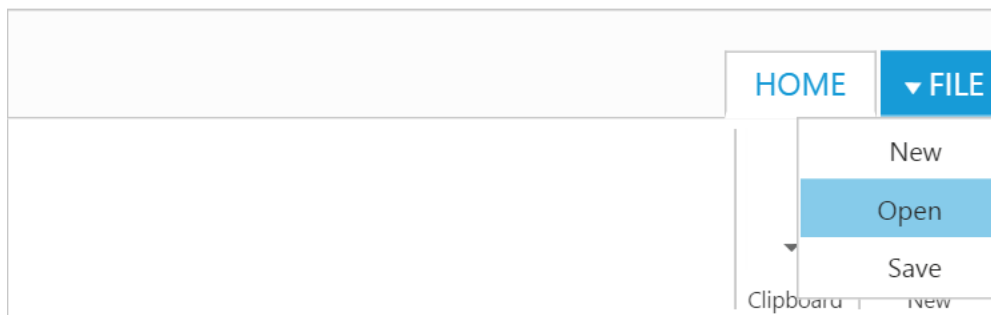
### HTML

```
<div id="defaultRibbon"></div>
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
<li><a>Save</a></li>
</ul>
</li>
</ul>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#defaultRibbon"), {
width: "40%",
enableRTL: true,
applicationTab: { type: ej.Ribbon.ApplicationTabType.Menu, menuItemID:
"ribbon", menuSettings: { openOnClick: false } },
tabs: [{
id: "home", text: "HOME", groups: [{
text: "New", alignType: ej.Ribbon.AlignType.Rows, content: [{
groups: [{
id: "new",
text: "New",
toolTip: "New",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
imagePosition: ej.ImagePosition.ImageTop,
prefixIcon: "e-icon e-ribbon e-new",
click: "executeAction"
}
}
}
],
defaults: {
type: ej.Ribbon.Type.Button,
width: 60,
height: 70
}
```

```

}
}]
},
{
text: "Clipboard", alignType: ej.Ribbon.AlignType.Columns, content: [{
groups: [{
id: "paste",
text: "paste",
toolTip: "Paste",
splitButtonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-ribbon e-ribbonpaste",
targetID: "pasteSplit",
buttonMode: "dropdown",
arrowPosition: ej.ArrowPosition.Bottom,
click: "executeAction"
}
}
],
defaults: {
type: ej.Ribbon.Type.SplitButton,
width: 50,
height: 70
}
},
]
},
]
});
});
}

```



## Appearance and Styling

### CssClass

When you want to display the **Ribbon** widget in a different style based on the appearance of your application, you can use this **cssClass** property to apply custom theme for the **Ribbon**. Specify a class name as the value for **cssClass** property. The specified class is added to the root element of the **Ribbon** widget. Now, you can easily override the styles of the **Ribbon** widget by accessing the styles from the root level (using the **cssClass** specified).

The following steps explain you on how to configure the **Ribbon** with custom theme using the **cssClass** property. Here, a class name “custom” is specified for the **cssClass**.

In an **HTML** page, specify the **<div>** elements to render the “Ribbon”.

### HTML

```
<div id="Ribbon"></div>
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
</ul>
</li>
</ul>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#defaultRibbon"), {
width: "500px",
cssClass: "custom",
applicationTab: {
type: ej.Ribbon.applicationTabType.menu,
menuItemID: "ribbon"
},
// tab item defined here
tabs: [{
id: "home",
text: "HOME",
// group with content & button settings
groups: [{
text: "New",
content: [{
groups: [{
id: "new",
text: "New",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-ribbon e-new",
}
}
}
}
}
}];
});
});
}
```

### CSS

```
<style>
```



```
.custom.e-js .e-header {  
background: #179bd7;  
}  
.custom.e-js .e-content {  
background: #ddd;  
}  
.custom .e-rbn-button.e-btn.e-select {  
background: #f5f5f5;  
color: #333;  
}  
</style>
```

Execute the above code example to render the following output.



### Themes

Ribbon control's style and appearance are controlled based on CSS classes and it has support of 12 default themes. You can check List of themes available for JavaScript's control which is mentioned in [this](#) page.

To apply styles refer to two files namely

`ej.widgets.core.min.css` `ej.theme.min.css`

When the file `ej.widgets.all.min.css/ej.web.all.min.css` is referred, it is not necessary to include the files `ej.widgets.core.min.css` and `ej.theme.min.css` in your project as `ej.widgets.all.min.css` is the combination of these two files.

### Customize Styles

Override default styles of Ribbon control by using its class name to customize it.

List of custom classes that are need to override styles. The following are specific class names you can use to modify styles and appearance in Ribbon.

*e-js* - used to represent element *ej-widget control*, that is present at the container element of Ribbon control. *e-header* - This class is applied to the header element of the Ribbon. *e-active-content* - This class is applied to the active content of the Ribbon. *e-expandcollapse* - This class is applied to the expand/collapse button in the Ribbon.

### Application Tab

- *e-apptab* - This class is applied to the application tab in the Ribbon.

### Tabs & Groups

*e-tab* - This class is applied to the Tabs in the Ribbon. *e-groupdiv* - This class is applied to the groups in the Ribbon. \* *e-active* - This class is applied to the active or selected tab in the Ribbon.

### Contextual Tabs

\* `e-contextlisset`, `e-contextual`, `e-contextualtabset` - This class is applied to the contextual tab and contextual tab set in the Ribbon.

### Custom Tooltip

\* `e-tooltipdiv`, `e-tooltiptitle`, `e-tooltipdesc`, `e-tooltiping` – Applies to custom tooltip and its icons settings.

### Gallery

`e-gallexpandcontent`, `e-gallerycontent` - Applies to gallery items and custom items for collapse and expanded state. `e-gallerymovediv`, `e-gallerybtn`, `e-moveupdiv`, `e-movedowndiv`, `e-expgallerydiv` - Applies to gallery item button and navigation buttons of gallery. \* `e-extracontent`, `e-galleryextrabtn`, `e-gallerymenu` - Applies to gallery custom items menu and button customization.

### Resizing

\* `e-resizebtn`, `e-resizediv` – Applies to resizing and its group button.

### Back Stage

`e-ribbonbackstagepage`, `e-ribbonbackstagetop`, `e-backstagetopicon` – Applies to back stage top header. `e-ribbonbackstagebody`, `e-backstageheader`, `e-backstagecontent`, `e-backstageli`, `e-backstageseparator`, `e-backstageactive` – Applies Backstage page side header and its contents.

### Quick Access Toolbar

- `e-rbnwithqat`- This class applied to the Quick access toolbar in the Ribbon.
- `e-rbnabove` - This class applied to the top Quick access toolbar.
- `e-qatooldiv`- This class applied to individual controls in the Quick access toolbar.
- `e-rbnbelow`- This class applied to the bottom Quick access toolbar.

### Load on Demand

Set [enableOnDemand](#) as true to enable load tab and backstage contents dynamically. Loading content on demand improves the initial rendering time of the ribbon by rendering tab and backstage content when tabs and backstage items are clicked.

### HTML

```
<div id="defaultRibbon"></div>
<div id="newCon">
  <table>
    <tr>
      <td>
        <button id="button" class="e-bsnewbtnstyle">Blank WorkBook</button>
      </td>
    </tr>
  </table>
</div>
<div id="accountCon">
  <div class="e-userDiv">
    <span>User Information</span>
  </div>
  <div class="e-accuser e-newpageicon"></div>
  <div class="e-userCon">
    <div>user</div>
  </div>
</div>
```

```

<div>any@syncfusion.com</div>
</div>
</div>
</div>
<a href="#">Sign out</a>
</div>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#defaultRibbon"), {
width: "50%",
enableOnDemand: true,
applicationTab: {
type: ej.Ribbon.ApplicationTabType.Backstage,
backstageSettings: {
text: "FILE",
height: 360,
width: 600,
headerWidth: 125,
pages: [{ id: "new", text: "New", contentID: "newCon" },
{ id: "close", text: "Close", enableSeparator: true, itemType:
ej.Ribbon.itemType.button },
{ id: "account", text: "Office Account", contentID: "accountCon"}]
}
},
tabs: [{
id: "home", text: "HOME", groups: [{
text: "Clipboard", alignType: ej.Ribbon.AlignType.Columns,
groupExpanderSettings: {
toolTip: "Clipboard"
}, content: [{
groups: [{
id: "paste",
text: "paste",
toolTip: "Paste",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-ribbon e-ribbonpaste"
}
}
]
},
],
defaults: {
type: ej.Ribbon.Type.Button,
isBig: true,
width: 50,
height: 70
}
}
}],
},
{
text: "New", alignType: ej.Ribbon.AlignType.Rows, content: [{
groups: [{
id: "new",
text: "New",
toolTip: "New",
buttonSettings: {

```

```

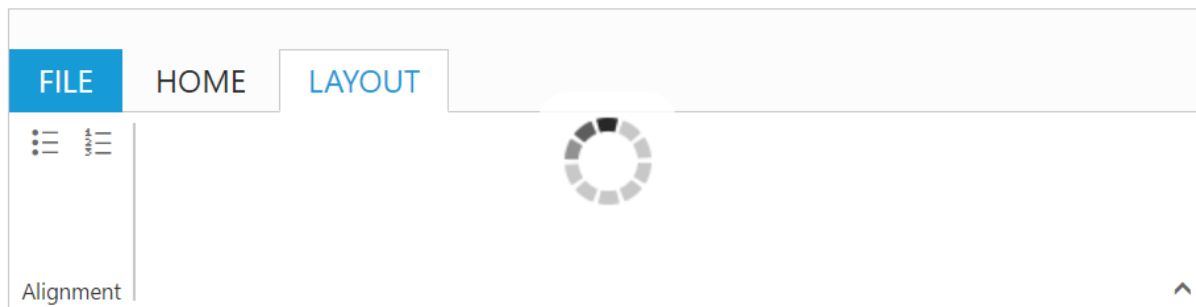
contentType: ej.ContentType.ImageOnly,
imagePosition: ej.ImagePosition.ImageTop,
prefixIcon: "e-icon e-ribbon e-new"
}
},
],
defaults: {
type: ej.Ribbon.Type.Button,
width: 60,
height: 70
}
}]
}]
},
{
id: "layout", text: "LAYOUT", groups: [{
text: "Alignment", alignType: ej.Ribbon.AlignType.Rows, content: [
{
groups: [{
id: "bullet",
text: "Bullet Format",
toolTip: "Bullets",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-ribbon e-bullet"
}
},
{
id: "number",
text: "Number Format",
toolTip: "Numbering",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-ribbon e-numbericon"
}
}],
defaults: {
type: ej.Ribbon.Type.Button,
isBig: false
}
}
}],
});
});
var buttonsample = new ej.Button($("#button"), {
size: "large",
height: 200,
width: 225,
contentType: "textandimage",
imagePosition: "imagetop",
prefixIcon: "e-icon e-blank e-infopageicon"
});
}
<style>
.e-accuser {

```

```

background-image: url("../content/ejthemes/common-
images/ribbon/User.jpg"),url("content/ejthemes/common-
images/ribbon/User.jpg");
}
.e-blank {
background-image: url("../content/ejthemes/common-
images/ribbon/blank.png"),url("content/ejthemes/common-
images/ribbon/blank.png");
}
.e-infopageicon {
background-repeat: no-repeat;
height: 150px;
width: 198px;
}
.e-newpageicon {
background-repeat: no-repeat;
height: 42px;
width: 42px;
}
.e-bspagestyle {
line-height: 0;
font-size: 30px;
}
.e-ribbon .e-ribbonbackstagepage .e-bsnewbtnstyle {
color: #212121;
background: #fdfdfd;
margin: 20px;
}
</style>

```



### Initially Collapsible

Set `'collapsible'` as true to render ribbon control in collapsed state, which results ribbon tabs to render without any content initially.

While using initially collapsible ribbon with `enableOnDemand` feature improves the performance by reducing initial loading time of ribbon.

### HTML

```

<div id="defaultRibbon"></div>
<div id="newCon">
<table>
<tr>
<td>
<button id="button" class="e-bsnewbtnstyle">Blank WorkBook</button>

```

```

</td>
</tr>
</table>
</div>
<div id="accountCon">
<div class="e-userDiv">
<span>User Information</span>
<div>
<div class="e-accuser e-newpageicon"></div>
<div class="e-userCon">
<div>user</div>
<div>any@syncfusion.com</div>
</div>
</div>
</div>
</div>
<a href="#">Sign out</a>
</div>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RibbonComponent {
$(function () {
var sample = new ej.Ribbon($("#defaultRibbon"), {
width: "50%",
enableOnDemand: true,
collapsible: true,
applicationTab: {
type: ej.Ribbon.ApplicationTabType.Backstage,
backstageSettings: {
text: "FILE",
height: 360,
width: 600,
headerWidth: 125,
pages: [{ id: "new", text: "New", contentID: "newCon" },
{ id: "close", text: "Close", enableSeparator: true, itemType:
ej.Ribbon.itemType.button },
{ id: "account", text: "Office Account", contentID: "accountCon"}]
}
},
tabs: [{
id: "home", text: "HOME", groups: [{
text: "Clipboard", alignType: ej.Ribbon.AlignType.Columns,
groupExpanderSettings: {
toolTip: "Clipboard"
}, content: [{
groups: [{
id: "paste",
text: "paste",
toolTip: "Paste",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-ribbon e-ribbonpaste"
}
}
]
},
],
defaults: {
type: ej.Ribbon.Type.Button,
isBig: true,

```

```

width: 50,
height: 70
}
}]
},
{
text: "New", alignType: ej.Ribbon.AlignType.Rows, content: [{
groups: [{
id: "new",
text: "New",
toolTip: "New",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
imagePosition: ej.ImagePosition.ImageTop,
prefixIcon: "e-icon e-ribbon e-new"
}
}
],
defaults: {
type: ej.Ribbon.Type.Button,
width: 60,
height: 70
}
}]
}],
{
id: "layout", text: "LAYOUT", groups: [{
text: "Alignment", alignType: ej.Ribbon.AlignType.Rows, content: [
{
groups: [{
id: "bullet",
text: "Bullet Format",
toolTip: "Bullets",
buttonSettings: {
contentType: ej.ContentType.ImageOnly,
prefixIcon: "e-icon e-ribbon e-bullet"
}
}
],
defaults: {
type: ej.Ribbon.Type.Button,
isBig: false
}
}
}],
defaults: {
type: ej.Ribbon.Type.Button,
isBig: false
}
}
}],
});
});

```

```

var buttonsample = new ej.Button($("#button"), {
  size: "large",
  height: 200,
  width: 225,
  contentType: "textandimage",
  imagePosition: "imagetop",
  prefixIcon: "e-icon e-blank e-infopageicon"
});
}
<style>
.e-accuser {
background-image: url("../content/ejthemes/common-
images/ribbon/User.jpg"),url("content/ejthemes/common-
images/ribbon/User.jpg");
}
.e-blank {
background-image: url("../content/ejthemes/common-
images/ribbon/blank.png"),url("content/ejthemes/common-
images/ribbon/blank.png");
}
.e-infopageicon {
background-repeat: no-repeat;
height: 150px;
width: 198px;
}
.e-newpageicon {
background-repeat: no-repeat;
height: 42px;
width: 42px;
}
.e-bspagestyle {
line-height: 0;
font-size: 30px;
}
.e-ribbon .e-ribbonbackstagepage .e-bsnewbtnstyle {
color: #212121;
background: #fdfdfd;
margin: 20px;
}
</style>

```

FILE

HOME

LAYOUT

## How to

### Get Ribbon object

After Ribbon initialization, Ribbon object is stored in a container element of Ribbon and it can be accessed for further processing.

### JAVASCRIPT

```

// "defaultRibbon" is Id of Ribbon control
$(function () {
var sample = new ej.Ribbon($("#defaultRibbon"), {

```



```
});  
});
```

## RichTextEditor

### Overview

**Rich text editor** is a component that help you to display or edit the content including tables, hyperlinks, paragraphs, lists, video, and images. The editor supports file and folder management using FileExplorer component.

### Key Features

- **Toolbar:** The editor's toolbar contains a collection of tools such as bold, italic and text alignment buttons that are used to format the content.
- **Content Area with customization:** The editor creates the iframe element as the content area on control initialization, it is used to display and editing the content.
- **Image and File browser:** The editor allows you to manage the images and files using FileExplorer. The FileExplorer enables you to insert images from online source as well as local computer where you want to insert the image in your content.
- **Links:** A hyperlink can be insert into the editor for quick access to the related information. The hyperlink itself can be a text or an image.
- **Tables:** The editor provides tools that make to work with tables in your content. You can add, edit, and remove the table as well as perform other table related tasks.
- **Localization:** The editor provides option to localize its strings, it is used to adapting the editor to a particular local language. By default, the editor will use the US English (en-US) as its language.
- **XHTML validation:** The editor provides option to validate its content through the enableXHTML property. When you set or modify the content into the editor, it continuously checks whether the HTML source of the content that you are creating is valid.

### Getting Started

This section helps to understand the getting started of RTE control with the step-by-step instruction.

#### Script and CSS reference

Create a new HTML file and include the below code

Add link to the CSS file from the specific theme folder to your HTML file within the head section. Refer the built-in available themes from [here](#).

Also add links to the [CDN](#) Script files along with the other external dependencies as depicted below,

#### HTML

```
<head>  
<meta charset="utf-8" />  
<title>Getting Started - RichTextEditor</title>  
<link href="http://cdn.syncfusion.com/{{ site.releaseversion  
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />  
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-  
1.10.2.min.js"></script>  
<script src="http://cdn.syncfusion.com/{{ site.releaseversion  
}}/js/web/ej.web.all.min.js"></script>  
</head>
```

---

**Note:** Uncompressed version of the required library files are available for the development or debugging purpose which can be generated from the custom script [here](#). Also to reduce the file size further please use [GZip](#) compression in your server.

---

### Control Initialization

Create a **TextArea** element within the body of the HTML document where the widget needs to be rendered.

#### HTML

```
<body>
<textarea id="texteditor"></textarea>
</body>
```

Initialize the editor in ts file by using the `ej.RTE` method.

#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {});
});
}
```

### Toolbar-Configuration

You can configure a toolbar with the tools as your application requires.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
toolsList: ["style", "lists", "doAction", "links", "images"],
tools: {
style: ["bold", "italic"],
lists: ["unorderedList", "orderedList"],
doAction: ["undo", "redo"],
links: ["createLink"],
images: ["image"]
}
});
});
}
```

### Setting and Getting Content

You can set the content of the editor as follows.

#### JAVASCRIPT

```

<textarea id="texteditor"></textarea>
<script type="text/javascript">
$("#texteditor").ejRTE({
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
});
</script>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
});
});
}

```

To retrieve the editor contents,

#### JAVASCRIPT

```
var currentValue = $("#texteditor").ejRTE("model.value");
```

You can find sample to quick start with the editor [here](#).

#### Toolbar Configuration

The editor's toolbar contains a collection of tools such as bold, italic and text alignment buttons that are used to format the content.

However, in most integrations, it's desirable to change the toolbar configuration to suit needs. Fortunately, that's quite easy to do too.

| Property                  | Description   |
|---------------------------|---|
| <a href="#">toolsList</a> | The toolsList option allows you to choose which tools appear on the toolbar, as well as the order and grouping of those items                       |
| <a href="#">tools</a>     | The toolsList property is used to get the root group order and tools property is used to get the inner order of the corresponding groups displayed. |

**Note:** By default, when you tab from the textbox to the RTE, the first tools in the Toolbar of RTE will get focus not in the text area. <BR>

But we can able to focus the RTE text area by setting the tab index attribute as -1 to avoid the focus on RTE toolbar when we tab from textbox to RTE - [Demo](#)

### Toolbar Items

The following table lists the available buttons and dropdowns on the toolbar:

| Name       | Summary   | Initialization  | IsDefault? |
|------------|---|---|------------|
| Font       | Applies font type, size, and color to the content.                            | tools: {<br>font: ["fontName", "fontSize", "fontColor", "backgroundColor"]<br>}             | No         |
| Font style | Applies bold, italic, underline, and strikethrough formatting to the content. | tools: {<br>style: ["bold", "italic", "underline", "strikethrough"]<br>}                    | Yes        |
| Alignment  | Align the content with left, center, and right margin.                        | tools: {<br>alignment: ["justifyLeft", "justifyCenter", "justifyRight", "justifyFull"]<br>} | Yes        |
| List       | Create a new list item (bulleted/numbered).                                   | tools: {<br>lists: ["unorderedList", "orderedList"]<br>}                                    | Yes        |
| Indents    | Allows to increase/decrease the indent level of the content.                  | tools: {<br>indenting: ["outdent", "indent"]<br>}   | Yes        |

|                  |   |   |     |
|------------------|---|---|-----|
| Undo/Redo Action | Allows to undo/redo the actions   | tools: {<br>doAction: ["undo", "redo"]<br>}   | Yes |
| Hyperlink        | Creates a hyperlink to a text or image to a specific location in the content. | tools: {<br>links: ["createLink", "removeLink"]<br>}  | Yes |
| Images           | Inserts an image from an online source or local computer.                     | tools: {<br>images: ["image"]<br>}  | Yes |
| Media            | Allows to embed a video into the document.                                    | tools: {<br>media: ["video"]<br>}   | Yes |
| Table            | Allows to add or modify Tables.   | tools: {<br>tables: ["createTable", "addRowAbove", "addRowBelow", "addColumnLeft", "addColumnRight", "deleteRow", "deleteColumn", "deleteTable"]<br>} | Yes |
| Casing           | Change the case of selected text in the content                               | tools: {<br>casing: ["upperCase", "lowerCase"]<br>}   | No  |
| Scripts          | Makes the selected text as superscript (higher) or subscript (lower).         | tools: {<br>effects: ["superscript", "subscript"]<br>}  | No  |

|                   |   |  |     |
|-------------------|---|--|-----|
| Format            | Clears the formatting options like bold, italic, underline and more.                | tools: {<br>formatStyle: ["format"]<br>}                   | Yes |
| Clipboard Actions | Controls the clipboard actions by applying specific action on the selected content. | tools: {<br>clipboard: ["cut", "copy", "paste"]<br>}       | No  |
| Edit              | Allows to make find and replace functionalities with the content.                   | tools: {<br>edit: ["findAndReplace"]<br>}                  | No  |
| view              | Allows to change the editor view mode.  | tools: {<br>view: ["fullScreen", "zoomIn", "zoomOut"]<br>} | No  |
| print             | Allows to print the editor content.   | tools: {<br>print: ["print"]<br>}                          | No  |

### Rearrange Group

The toolbar contains groups, which are similar or related functionalities of toolbar items for efficient access. By default, the groups are arranged using the following order

#### JAVASCRIPT

```
toolsList: ["formatStyle", "font", "style", "effects", "alignment", "lists", "indenting", "clipboard", "doAction", "clear", "links", "images", "media", "tables", "casing", "customTools", "view"]
```

The group can be rearranged on customization using the [toolsList](#) property.

**HTML**

```

<textarea id="editor"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
toolsList: ["links", "lists", "doAction", "style", "images"],
tools: {
style: ["bold", "italic"],
lists: ["unorderedList", "orderedList"],
doAction: ["undo", "redo"],
links: ["createLink", "removeLink"],
images: ["image"]
}
});
});
}

```

**Note:** If you are not specify any group in **toolsList** property, the editor will create the toolbar with default group.

**Undo and Redo**

Undo and Redo buttons allow you to editing the text by disregard/cancel the recently made changes and restore it to previous state. It is a useful tool to restore the performed action which got changed by mistake. Up to 50 actions can be undo/redo in the editor by default.

To undo and redo operations, do one of the following:

- Press the undo/redo button on the toolbar
- Press the **Ctrl + Z**/**Ctrl + Y** combination on the keyboard

**HTML**

```

<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
toolsList: ["doAction"],
tools: {
doAction: ["undo", "redo"]
}
});
});
}

```

### Clipboard Operations

The editor provides support for the clipboard operations (cut, copy, and paste) in all text and images using the toolbar buttons and the keyboard shortcuts. Toolbar includes buttons through which the clipboard operations, such as Cut, Copy, and Paste can be accessed.

You can use the keyboard shortcuts to perform the clipboard operations.

- Cut - CTRL+X
- Copy - CTRL+C
- Paste - CTRL+V

---

**Note:** Some browsers block the clipboard access from JavaScript. If you want to use the Cut, Copy, and Paste buttons on the toolbar, you need to allow JavaScript to use the clipboard. If you don't want to do this configuration, use CTRL+C, CTRL+X, and CTRL+V keyboard commands.

---

### HTML

```
<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
toolsList: ["clipboard"],
tools: {
clipboard: ["cut", "copy", "paste"]
}
});
});
}
```

### Types of responsive toolbar

Two types of toolbar modes are available for RTE in responsive mode. This includes "inline" and "popup". Toolbar Type can be set through API [toolbarOverflowMode](#) whose default value is "popup".

### HTML

```
<textarea id="editor"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
isResponsive:true,
toolbarOverflowMode:"inline",
});
});
}
```



**Note:** If you are not specifying toolbarOverflowMode, responsive RTE will be rendered with default popup toolbar.

## Context Menu

Editor provides custom context menu support, which enables more interaction on the content modification and also it can be enabled dynamically. The showContextMenu property helps to enable custom context menu within editor area.

Based on the target content type context menu provides different actions. Refer the details with below table.

### HTML

```
<textarea id="rteSample">
<p><b>Description:</b></p>
<p>The Rich Text Editor (RTE) control is easy to render in the
client side. Customers can easily edit the contents and get the HTML
content for
the displayed content. A rich text editor control provides users with a
toolbar
that helps them to apply rich text formats to the text entered in the
text
area. </p></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
showContextMenu: true
});
});
}
```

- Based on the target content type context menu provides different actions- refer the details with below table.

| Content-Type  | Supported Actions  |
|---------------|--|
| Text content  | cut, copy, paste, add/edit/open/remove hyperlink.  |
| Image content | cut, copy, paste, image properties.  |
| Table content | cut, copy, paste, insert row/column, remove row/column/table, edit table properties, add/edit/open/remove hyperlink. |

**Note:** We have given support to own context menu by restricting the default browser context menu, which provides you the options for quick access but, with that clipboard action are restricted based on browser behavior. <BR>

However we can disable the context menu by using ShowContextMenu API and it needs to be set as false, if you wish to continue with default browser context menu.

## Working with Content

The editor creates the iframe element as the content area on control initialization, it is used to display and editing the content. In Content Area, the editor displays only the body tag of a <iframe> document. To set or modify details in the <head> tag, use [Source view](#) of the editor.

### Iframe Attributes

The editor allows you to passing an additional attributes to body tag of a <iframe> element using [iframeAttributes](#) property. The property contains name/value pairs in string format, it is used to override the default appearance of the content area.

**Note:** the content area's font, color, margins, and background can be overridden using [iframeAttributes](#) property. You can specifies the editable behavior (content editable) of the content also in this property.

### HTML

```
<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
iframeAttributes:{style: "background-color:#e0ffff;color:#6495ed;"}
});
});
}
```

**Note:** Background image for the RTE control : [Link](#) <BR>

Set default font for the Iframe : [Link](#)

### Content Editable

The editor provides option to control the editable behavior using [allowEditing](#) property. When the allowEditing property is set to false, the editor disables its editing functionality.

### HTML

```
<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
allowEditing: false
});
});
}
```

```
</script>
```

The contentEditable attribute allows you to make any element of HTML content to become editable or non-editable.

### HTML

```
<textarea id="editor"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"<p>The RichTextEditor (RTE) control enables you to edit the contents with
h insert table and images,</p>" +
"<p> it also provides a toolbar that helps to apply rich text formats to
the content entered in the TextArea.</p>",
});
var editor = $("#editor").ejRTE("instance");
var iframeDoc = editor.getDocument();
var paragraph = $("p", iframeDoc.body);
$($ (paragraph)[1]).attr("contenteditable", "false");
});
});
}
```

**Note:** Content editable is fully compatible with latest browsers, to know more details, see [here](#).

### Submit Content

The editor allows you to process its content before it is being submitted to the server on form submit event. You can use this option to validate content on the client side to prevent invalid data from being submitted to the server.

This example shows how to encode the HTML content before form submit event.

### HTML

```
<form>
<textarea id="rteSample"></textarea>
<button type="submit">Submit</button>
</form>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
});
$("form").on("submit", function () {
var editor = $("#editor").data("ejRTE");
var encoded = $('<div />').text(editor.model.value).html();
```

```

$("#editor").val(encoded);
});
});
}

```

### Persistence

The editor is capable to persist its content with HTML format. By default, the persistence support is disabled in the editor. When you set the [enablePersistence](#) property to true, the persistence will be enabled in the editor.

**Note:** [local storage](#) is not supported below ie9 version, therefore persistence support is fallback to [cookie](#).

### HTML

```

<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
enablePersistence: true
});
});
}

```

### Apply Font color and Background color

If you want to apply font color or background color for a selected content of RTE you can use font color and background color tools. These tools contains a color palette with basic colors along with an option called **"More colors.."** in order to choose custom colors from color picker dialog. You can apply transparent background color for selected text through **transparent** button available in background color palette.

### HTML

```

<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images, it also provides a toolbar that helps to apply ri
ch text formats to the content entered in the TextArea.",
tools: {
font: ["fontName", "fontSize", "fontColor", "backgroundColor"]
}
});
});
}

```

### Insert the content at cursor

If you want to insert/paste the content at the current cursor position (or) to replace the selected content with some formatting, you can use pasteContent method in the editor.

#### HTML

```
<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value: "The RichTextEditor (RTE) control enables you to edit the contents
with insert table and images, "+
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
});
});
}
function pasteContent() {
var selectedHtml = sample.getSelectedHtml();
sample.pasteContent("<p style ='background-color:yellow;color:skyblue'>"
+ selectedHtml + "</p>");
}
```

### Working with Selection

The editor control provides option to select all the content and in addition to selection of a particular range of content.

#### Select All

The [selectAll](#) method enables you to select the entire content including images in the editor by programmatically.

**Note:** the selection highlight is invisible if the editor does not have focus. So, if you want to call the selectAll method, focus the editor before.

#### HTML

```
<textarea id="rteSample"></textarea>
<button onclick="selectAll()">Select All</button>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images, " +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
});
});
}
function selectAll() {
sample.selectAll();
}
```

```
</script>
```

### Select a Range

You can programmatically select a range of content in the editor using the `selectRange` method. To select a range, create a range object with desired offset position and pass it as arguments to `selectRange` method. The range object is created from `createRange` method.

#### HTML

```
<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value: "<ul>" +
"<li>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</li>" +
"<li>Aliquam tincidunt mauris eu risus.</li>" +
"<li>Vestibulum auctor dapibus neque.</li>" + "</ul>"
});
range = sample.createRange();
var liTag = $(sample.getDocument().body).find("li");
if (!sample._isIE8()) {
range.setStart(liTag[1], 0);
range.setEnd(liTag[2], 1);
}
else {
range = sample.getDocument().body.createTextRange();
range.moveToElementText(liTag[2]);
}
sample.selectRange(range);
});
}
```

### Get Selection

The following public methods helps you to retrieve the selected content from the editor:

- [getText](#) method is used to get the currently selected content as raw text.
- [getSelectedHtml](#) method is used to get the HTML source of currently selected content.

#### HTML

```
<textarea id="rteSample"></textarea>
<button onclick="getSelection()">get Selection</button>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
});
});
}
```

```
});  
});  
}  
function getSelection() {  
var selectedText = sample.getText();  
var selectedHtml = sample.getSelectedHtml();  
}
```

## Working with Tables

The editor provides tools that make to work with tables in your content. You can add, edit, and remove the table as well as perform other table related tasks.

### Create a Table

By default, [Insert Table](#) tool is enabled in the editor's toolbar. There are two ways to insert a table into the editor's content.

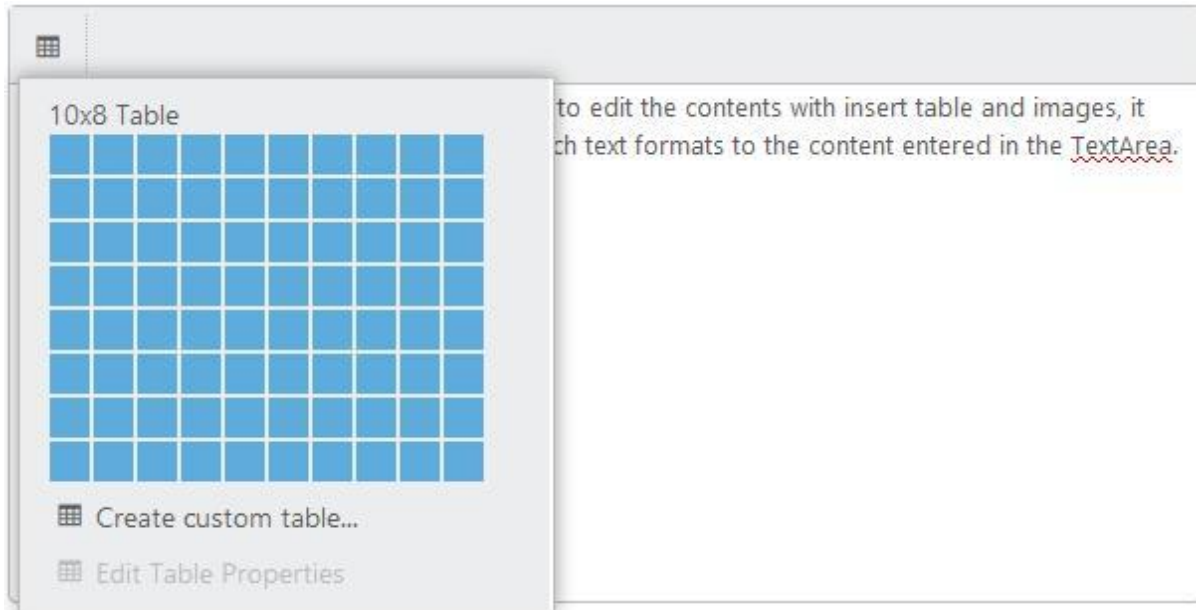
- [Insert a table](#)
- [Insert a custom table](#)

## HTML

```
<textarea id="rteSample"></textarea>  
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module RTEComponent {  
$(function () {  
var sample = new ej.RTE($("#rteSample"), {  
value:  
"The RichTextEditor (RTE) control enables you to edit the contents with i  
nsert table and images," +  
" it also provides a toolbar that helps to apply rich text formats to the  
content entered in the TextArea.",  
toolsList: ["tables"],  
tools: {  
tables: ["createTable", "addRowAbove", "addRowBelow", "addColumnLeft",  
"addColumnRight", "deleteRow", "deleteColumn", "deleteTable"]  
}  
});  
});  
}
```

### Insert a Table

You can insert a basic table by select the "Insert Table" tool from toolbar and drag the cursor over the grid until you highlight the number of columns and rows you want. A table can be inserted as large as with ten columns and eight rows (10 \* 8 cells). When you finish inserting tables, click in a cell and start typing or insert an image.



**Note:** If you want to make any adjustment with drawn table, you can [add/remove rows and columns](#).

#### Custom Table

You can use custom table tools to insert tables with custom behaviors. You can create a table with more than ten columns and eight rows, as well as set the table related attributes (such as width, height, cell spacing/padding, and more) using the custom table dialog.

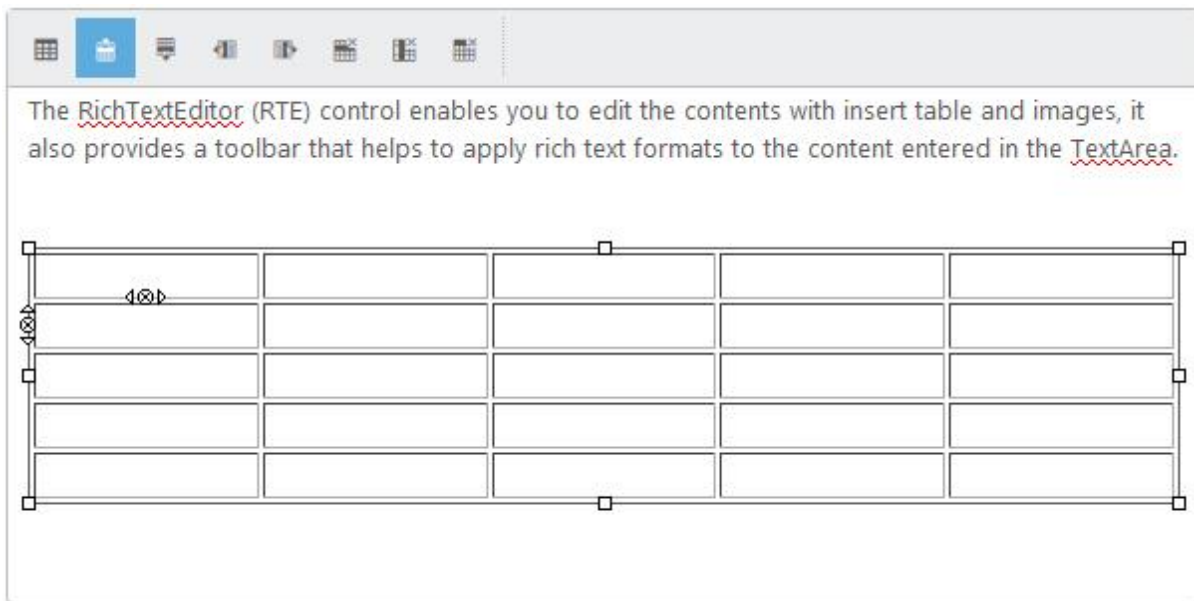
A screenshot of the "Create custom table..." dialog box. The dialog has a blue header bar with the title "Create custom table..." and a close button (X). The main area contains several input fields and controls: "No.of Columns" with a value of 3 and a spinner, "No.of Rows" with a value of 3 and a spinner, "Width" and "Height" with empty text boxes, "Cell spacing" and "Cell padding" with empty text boxes, "Border" with a dropdown menu, "Caption" with a checkbox, and "Alignment" with a dropdown menu. At the bottom right, there are "Insert" and "Cancel" buttons.



### Insert and Delete a Row or Column

You can choose the following options to modify the inserted table from the toolbar, it will be enabled when you're in table cell.

- Add a row above
- Insert row below
- Insert column left
- Insert column right
- Delete a column
- Delete a row

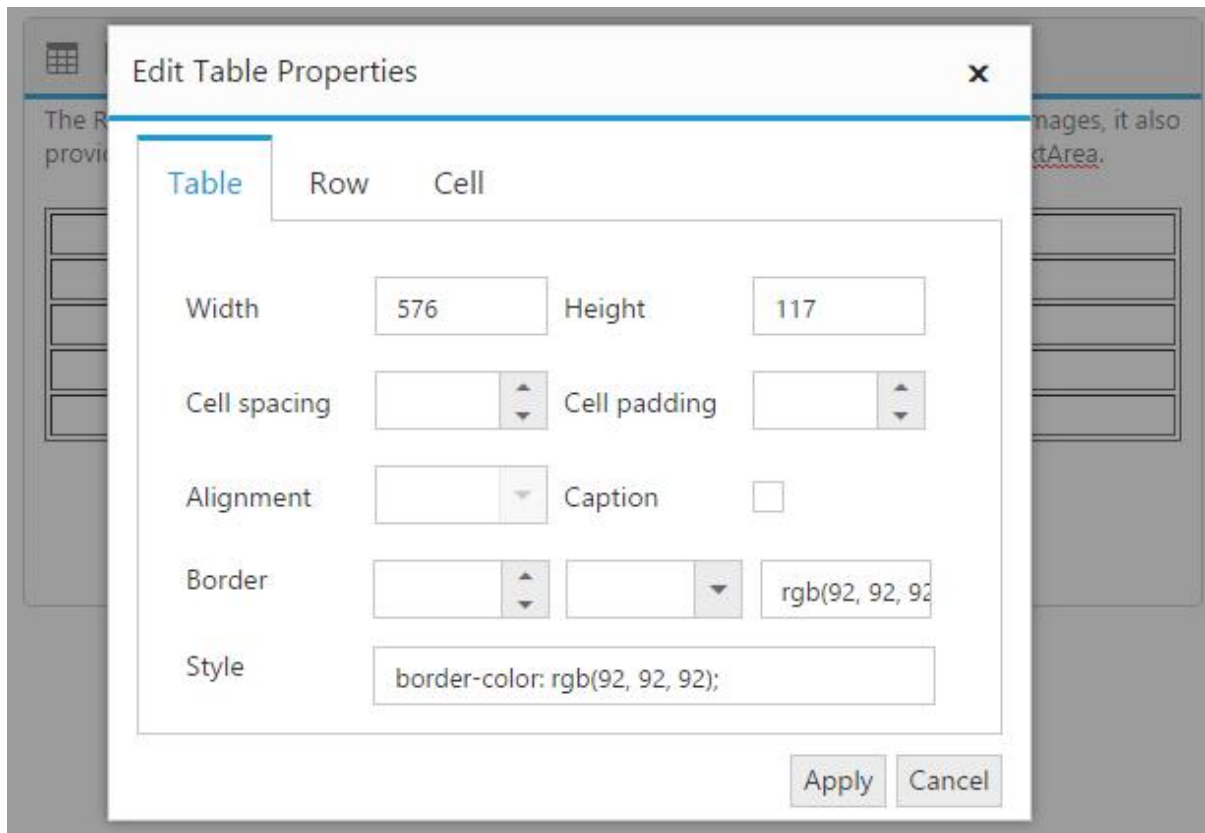


**Note:** You can also quickly add a new row by clicking on the lower-right cell of the table and pressing the Tab key.

### Format a Table

After you create a table, you can format the entire table by using **Table Properties** dialog. In the Table Properties dialog, set or modify each table styles to apply the styles to table elements.

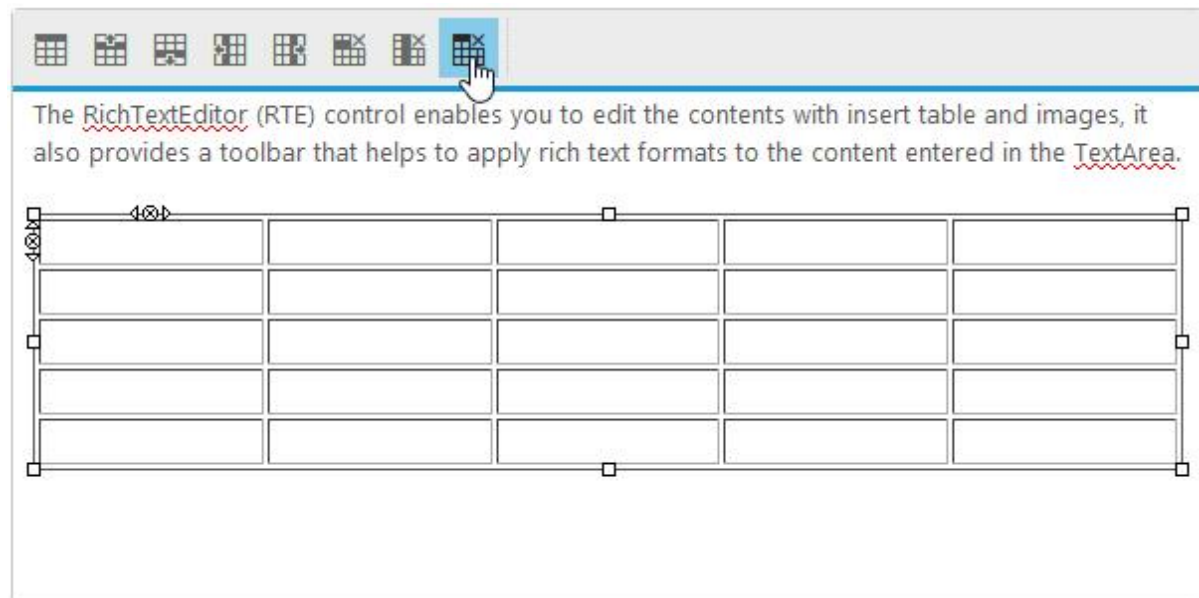
- Table - width, height, style, border, and alignment.
- Cell - border, spacing, and padding.
- Row - height, border, and text align.



**Note:** When you click in a table cell, the **Table Properties** option will be enabled under **Table** tool on the editor's toolbar.

#### Delete a table

To delete a table from your content, focus on the table and select "Delete a Table" tool from toolbar. It will delete the entire table with all formatting content.



---

**Note:** To delete the table contents alone, select the required content of the table which you want to delete and press Delete key. All the content disappears but the rows and columns remain along with its formatting.

---

### Working with Hyperlinks

A hyperlink can be insert into the editor for quick access to the related information. The hyperlink itself can be a text or an image.

#### Add and Edit a hyperlink

To add a hyperlink to the editor, follow these instructions:

1. Point the cursor anywhere within the editor where you'd like to insert the link. It is also possible to select a text or an image within the editor and can be converted to the hyperlink.
2. Click the "Insert/Edit hyperlink" tool on the Toolbar.
3. In the **Web Address** box, type or paste the destination for the link you are creating.
4. In the **Text** box, type or edit the required text that you want to display text for the link.

---

**Note:** Leave this textbox blank if you want to display the full link as hyperlink in the editor.

---

5. To display additional helpful information when you place the pointer on the hyperlink, type the required text in the "Tooltip" field.
6. Click OK button to insert a hyperlink.

### HTML

```
<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
toolsList: ["links"],
tools: {
links: ["createLink", "removeLink"]
}
});
});
}
```

---

**Note:** hyperlinks on a picture are not always visible, but the pointer's appearance will be changed on positioning the mouse pointer over it.

---

#### Remove a hyperlink

If you want to remove a hyperlink from a text or image, select the text or image with the hyperlink and click "Remove Hyperlink" tool from toolbar. It will keep the text or image.

## Image and File browser

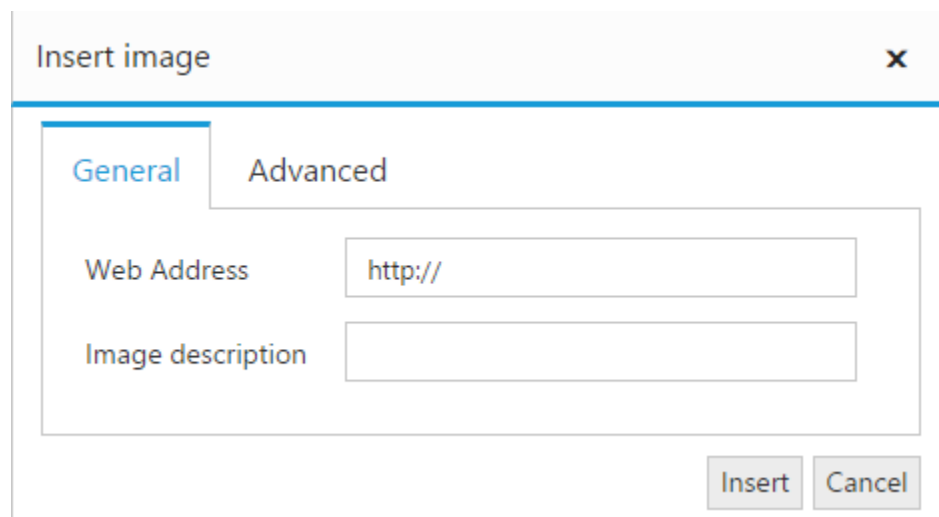
The editor allows you to manage the images and files using **FileExplorer**. The FileExplorer enables you to insert images from online source as well as local computer where you want to insert the image in your content. The Image and file browser is the ability to upload pictures and link file to the editor.

### Insert a Image from Online Source

If you want to insert an image from online source like Google, Ping, etc., you need to enable images tool on the editor's toolbar. By default, the images tool is open a simple dialog which allows you to inserting an image from online source.

#### HTML

```
<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
toolsList: ["images"],
tools: {
images: ["image"]
}
});
});
}
```



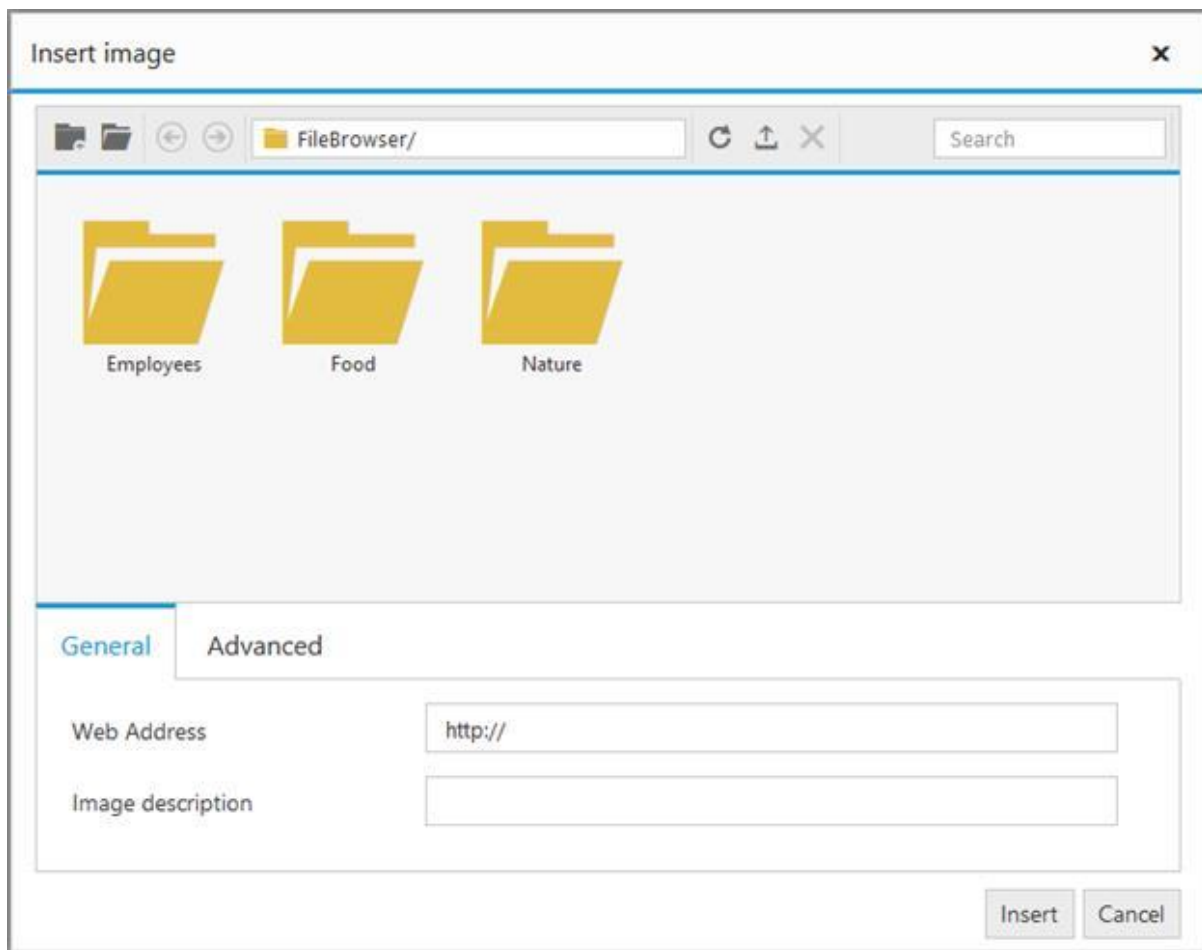
### Insert a Image from Your Computer

Configure the imageBrowser and fileBrowser property to insert an image from your computer. You can specify the settings required by the FileExplorer for create, read, upload, and destroy the files and images from the explorer.

#### HTML

```
<textarea id="rteSample"></textarea>
var fileService =
"http://mvc.syncfusion.com/OdataServices/fileExplorer/fileoperation/doJSO
NAction";
```

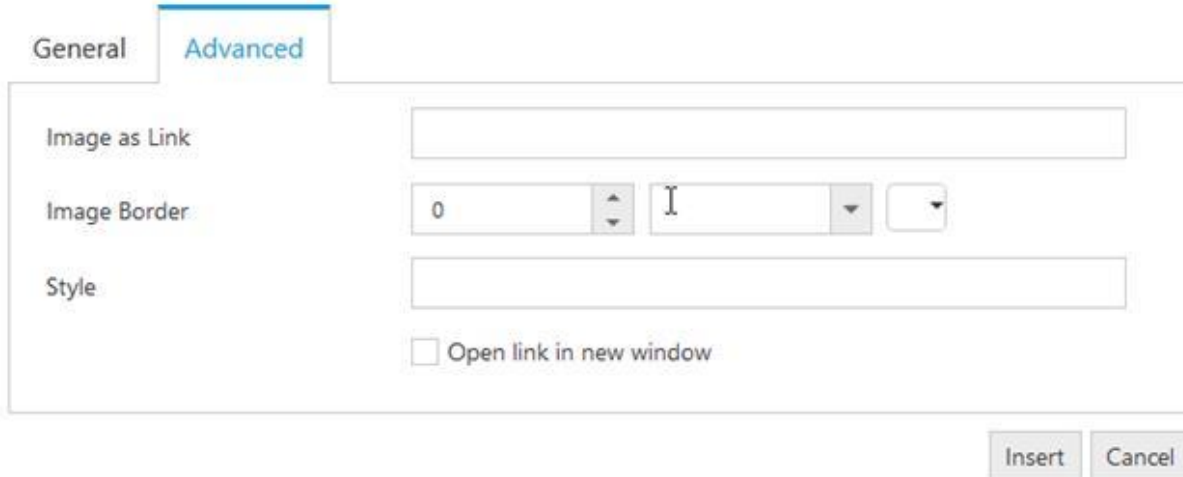
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
toolsList: ["images"],
tools: { images: ["image"] },
imageBrowser: {
filePath: "http://mvc.syncfusion.com/OdataServices/FileBrowser/",
ajaxAction: fileService,
extensionAllow: "*.png, *.gif, *.jpg, *.jpeg"
},
fileBrowser: {
filePath: "http://mvc.syncfusion.com/OdataServices/FileBrowser/",
ajaxAction: fileService,
extensionAllow: "*.txt, *.pdf"
}
});
});
}
```



**Note:** FileExplorer component has been implemented and integrated with the editor in Volume 1, 2015 release. For more information about FileExplorer component, see [here](#).

### Image Properties

You can set or modify properties of an image using the image dialog. It allows you to add links to images, apply border and additional styles. The editor provides option to specify the alternate text for an image, if the image cannot be displayed.



The image shows a dialog box with two tabs: 'General' and 'Advanced'. The 'Advanced' tab is selected. It contains the following fields and controls:

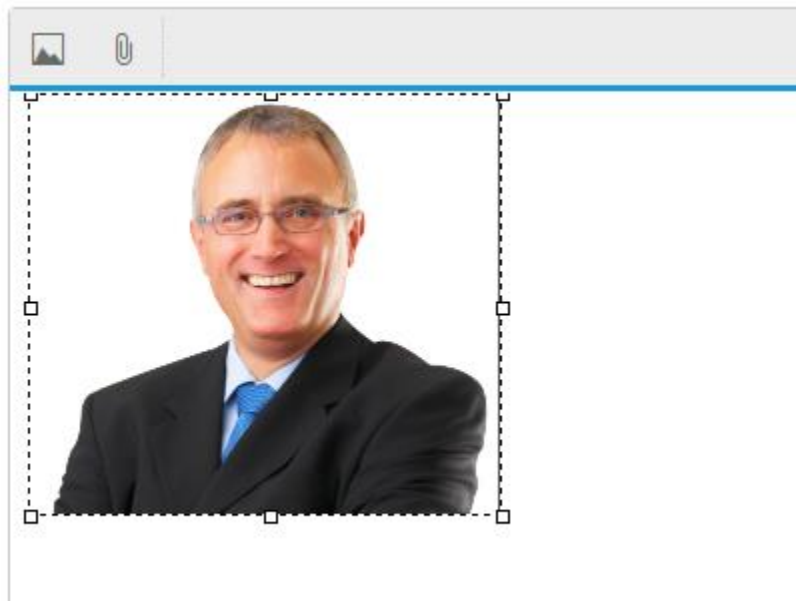
- Image as Link:** A text input field.
- Image Border:** A numeric input field with the value '0', a vertical spinner, a text input field with the character 'I', a dropdown arrow, and a small square button.
- Style:** A text input field.
- Open link in new window:** An unchecked checkbox.
- Buttons:** 'Insert' and 'Cancel' buttons at the bottom right.

### Resize an Image

You can able to resize an image either manually or set the width and height in the image dialog.

#### Resize Manually

You can resize an image by manually select an image, and drag a handle until the image is the desired size.



---

**Note:** Set the default height and width of the Images which was inserted into the RTE text area in “change” event of RTE - [Link](#)

---

### Set Width and Height

The editor provides you to set the width and height properties to change the size of an image (rather than forcing you to set in style attributes) using [showDimensions](#) property. By default, the Constrain Proportion checkbox is selected to resize an image to an exact proportion. To apply the exact width and height that you specify into the Height and Width textboxes, uncheck the Constrain Proportions checkbox.

### HTML

```
<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
toolsList: ["images"],
tools: { images: ["image"] },
showDimensions: true
});
});
}
</script>
```

The screenshot shows the RTE Image browser dialog with the 'General' tab selected. It contains the following elements:

- Web Address:** A text input field.
- Image description:** A text input field.
- Dimensions:** Two text input fields for width and height, separated by an 'X'.
- Constrain Proportions:** A checked checkbox.
- Buttons:** 'Insert' and 'Cancel' buttons at the bottom right.

### Suppression of the Image Browser

The General and Advanced tabs in the RTE Image browser can be removed by setting its corresponding display CSS property to none.

### CSS

```
<style type="text/css" class="cssStyles">
div.e-rte-imageTab.e-tab.e-js.e-widget {
display: none;
}
</style>
```

Can remove the Add **NewFolder** button by using [removeToolbarItem](#) property of Image Browser in the RTE create event.

### HTML

```
<textarea id="rteSample" rows="10" cols="30" style="width: 740px; height: 440px">
```

Description:

The Rich Text Editor (RTE) control is an easy to render in client side. Customer easy to edit the contents and get the HTML content for the displayed content. A rich text editor control provides users with a toolbar that helps them to apply rich text formats to the text entered in the text area.

```
</textarea>
```

```
var sample = new ej.RTE($("#rteSample"), {
  toolsList: ["images"],
  tools: { images: ["image"] },
  width: "100%",
  minWidth: "10px",
  imageBrowser: { filePath:
    "http://mvc.syncfusion.com/OdataServices/FileBrowser/", ajaxAction:
    fileService, extensionAllow: "*.png, *.gif, *.jpg, *.jpeg, *.docx",
    ajaxSettings: ajaxSettings },
  fileBrowser: { filePath:
    "http://mvc.syncfusion.com/OdataServices/FileBrowser/", ajaxAction:
    fileService, extensionAllow: "*.txt, *.png, *.pdf, *.jpeg", ajaxSettings:
    ajaxSettings1 },
  create: function (args) {
    this._explorerObj.removeToolbarItem("NewFolder");
  }
});
```

## Find and Replace

RTE provides find and replace support, which is used to search for a keyword in RTE content and replace the matched keyword with a specified text. In order to use it, we have to enable the find and replace item in the editor toolbar (or) Press CTRL+F key.

## HTML

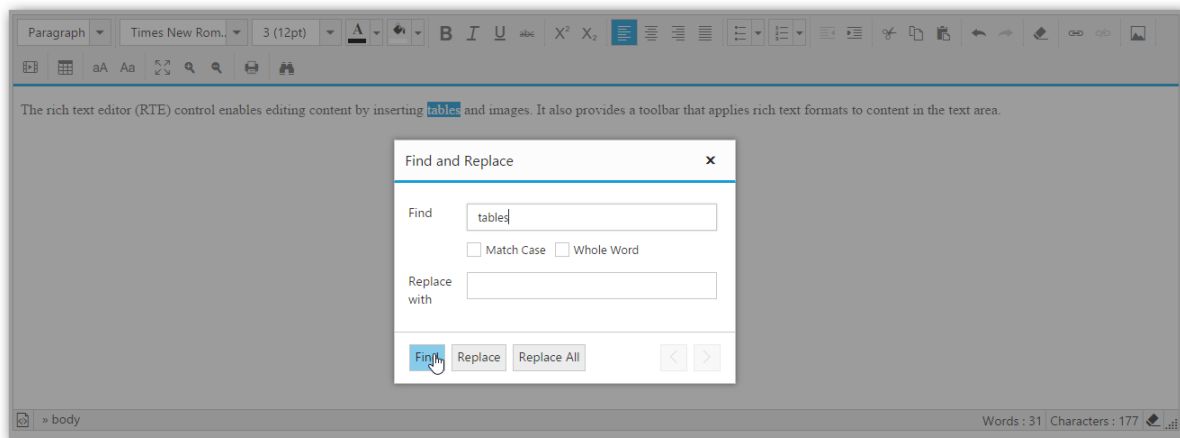
```
<textarea id="rteSample">
<p><b>Description:</b></p>
<p>The Rich Text Editor (RTE) control is easy to render in the
client side. Customers can easily edit the contents and get the HTML
content for
the displayed content. A rich text editor control provides users with a
toolbar
that helps them to apply rich text formats to the text entered in the
text
area. </p></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
tools: {edit: ["findAndReplace"]}
});
});
}
```



- Find and Replace action should happen in a following sequence:

Find => Replace (or) Replace All.

| Action     | Descriptions  |
|------------|---|
| Find       | Finds a keyword matches with the editor content.it consist of following filters. <i>Match Case</i> .<br>Whole Word. |
| Replace    | Replaces the particular selected match with the specified text.   |
| ReplaceAll | Replaces the entire matches with the specified text.  |



**Note:** Before performing the Replace/ReplaceAll action, we must do the find action to validate the match's availability.

## Footer

This option allows you to specify which footer elements should display at the bottom of the editor. The available footer elements are listed below:

1. HTML View
2. HTML Tag Info
3. Characters Count / Word Count
4. Clear Format
5. Resizer

## HTML

```
<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
showFooter: true
```

```

});
});
}

```

### Source View

RichTextBox includes the ability for users to directly edit HTML code via “Source View” in a separate dialog. If you made any modification in Source view directly, click Update button to synchronize with Design view. This provides power users with more flexibility over the content they create.

You can paste HTML text into Source view. If you cut or copy from HTML source such as page source of Browser, paste as HTML (without escape characters) into Source view.

### HTML

```

<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
showFooter:true,
showHtmlSource:true
});
});
}

```

---

**Note:** Source view is useful for working directly with raw HTML text, so this tool is mainly used for advanced users who would like to have more control over the source of their content.

---

### HTML Tag Info

The HTML tag info tool that shows the path of currently selected tag along with hierarchy of parent tags to which it belongs. The tag information is displayed at the bottom of the editor. It is used to determine which element has the focus in the editor's content.

### HTML

```

<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
showFooter:true,
showHtmlTagInfo:true
});
});
}

```

---

**Note:** The outermost tag is the body tag of < iframe > element in design view, so it shows the path from currently selected path to the body tag.

---

### Characters Count/Word Count

The editor automatically counts the number of characters and words in the content while you type. The characters and words count displayed at the bottom of the editor. You can limit the number of

characters in your content using [maxLength](#) property. By default, the editor sets the characters limit value as 7000 characters.

### HTML

```
<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
showFooter: true,
showWordCount: true,
showCharCount: true,
maxLength: 500
});
});
}
```

**Note:** The editor counts the characters by including the space, and this validation occurs while pasting the content into the editor also.

### Clear Format

The clear format tool is useful to remove all formatting styles (such as bold, italic, underline, color, superscript, subscript, and more) from currently selected text. As a result, all the text formatting will be cleared and return to its default formatting styles. When you set the [showClearFormat](#) property to true, the clear format tool will be displayed at bottom of the editor.

### JAVASCRIPT

```
<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
showFooter: true,
showClearFormat: true
});
});
}
```

### Resize Handle

When you set the [enableResize](#) property to true, resize handle will be displayed at bottom-right corner of the editor. You can drag the handle to change its size. On resizing, the editor will automatically adjust the toolbar, content area, and footer within it accordingly. Resize limits can be defined via [minHeight](#), [maxHeight](#), [minWidth](#), and [maxWidth](#) properties. You can specify the size of the editor programmatically through the [height](#) and [width](#) properties.

### HTML

```
<textarea id="editor"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
```

```
$(function () {
  var sample = new ej.RTE($("#rteSample"), {
    showFooter:true,
    enableResize:true,
    width:600,minWidth:250,maxWidth:750,
    height:300,minHeight:250,maxHeight:500
  });
});
```

**Note:** 1. As resizable option will be added in the footer of RTE, so set the showFooter property also as "true". <BR>

2. In order to showcase only the resizer, disable the other properties in the Footer such as showClearFormat, showClearFormat, showCharCount, showWordCount

3. When you set the enableRTL property to true, the resize handle will automatically positioned to the bottom-left corner of the editor.

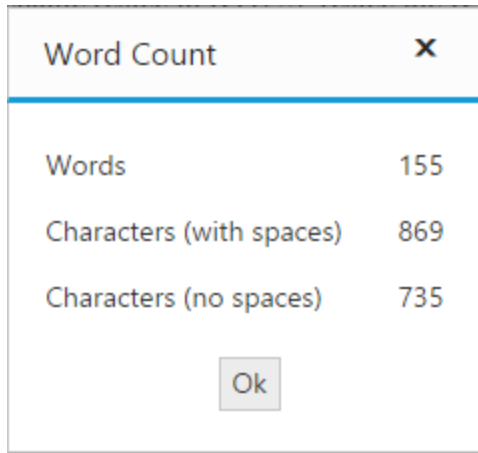
### Characters Count/Word Count

The editor automatically counts the number of characters and words in the content while you type. The characters and words count displayed at the bottom of the editor. You can limit the number of characters in your content using [maxLength](#) property. By default, the editor sets the characters limit value as 7000 characters.

### HTML

```
<textarea id="editor"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
  $(function () {
    var sample = new ej.RTE($("#rteSample"), {
      showFooter: true,
      showWordCount: true,
      showCharCount: true,
      maxLength: 500
    });
  });
}
```

By clicking the Characters Count/Word Count labels in footer , The word and character count information dialog is opened. It contains the details of the number of words and characters with and without spacing.



**Note:** The editor counts the characters by including the space, and this validation occurs while pasting the content into the editor also.

### Validation

You can validate the RichTextEditor's value on form submission by applying [validationRules](#) and [validationMessage](#) to the RichTextEditor.

**Note:** [jquery.validate.min](#) script file should be referred for validation, for more details, refer [here](#).

### jQuery Validation Methods

The following are jQuery validation methods.

#### List of jQuery validation methods

| Rules        | Description   |
|--------------|---|
| required     | Requires value for the RichTextEditor control.              |
| minWordCount | Requires the value to be of given minimum words count.      |
| minlength    | Requires the value to be of given minimum characters count. |
| maxlength    | Requires the value to be of given maximum characters count. |

### Validation Rules

The validation rules help you to verify the content by adding validation attributes to the text area. This can be set by using [validationRules](#) property.

### Validation Messages

You can set your own custom error message by using [validationMessage](#) property. To display the error message, specify the corresponding annotation attribute followed by the message to display.

**Note:** jQuery predefined error messages to that annotation attribute will be shown when this property is not defined. The below given example explain this behavior of 'maxLength' attribute,

When you initialize the RichTextEditor widget, it creates a text area hidden element which is used to store the value. Hence, the validation is performed based on the value stored in this hidden element.

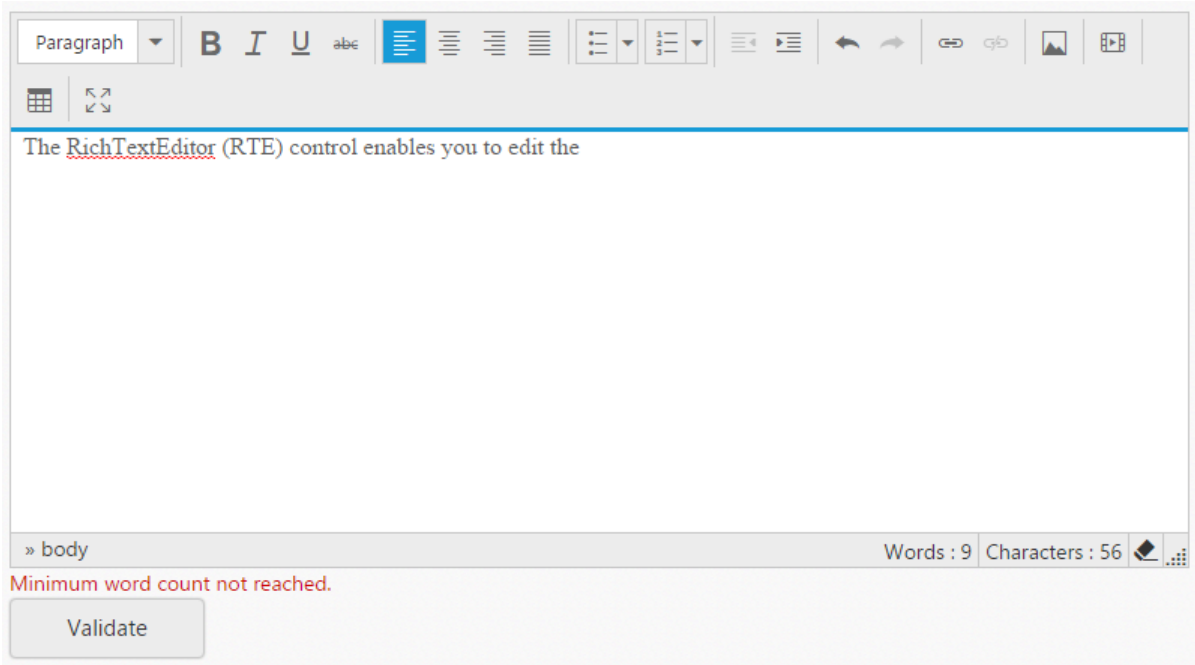
Required field, minlength, maxlength, minWordCount and maxWordCount values validation is demonstrated in the below given example.

### HTML

```
<form id="form1">
<textarea id="rteSample"></textarea>
<br/>
<button id="save">Validate</button>
</form>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.Button($("#save"), {
size: "large",
showRoundedCorner: true,
type : "submit"
});
var sample1 = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
allowEditing: true,
showFooter : true,
validationRules: {
required: true,
minlength:15,
maxlength: 150,
minWordCount: 10,
maxWordCount:50,
},
validationMessage: {
required: "Required RTE value",
minWordCount: "Minimum word count not reached.",
maxWordCount: "Maximum word count reached."
}
});
});
}
```



## XHTML Validation

The editor provides option to validate its content through the [enableXHTML](#) property. When you set or modify the content into the editor, it continuously checks whether the HTML source of the content that you are creating is valid. The editor examines the HTML markup and then removes the elements or attributes that are not valid.

## HTML

```
<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
enableXHTML: true
});
});
}
```

The editor checks the following settings on validation:

- **Attributes**
- Must be specified in lowercase
- Proper use of quotation marks around the attributes
- Must be valid attributes for corresponding HTML element
- All the required attributes must be included in the HTML element

- Accepts the allowed values alone such as true or false.
- **HTML Elements**
- Must be in lowercase
- All opening tags must be closed
- Allows only the valid HTML elements

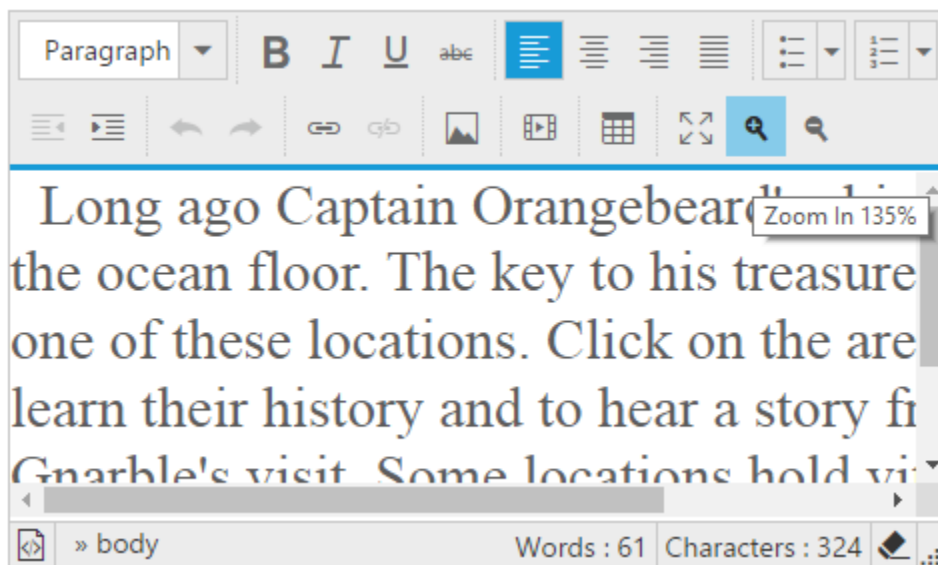
## Zoom

The editor provides zoom tools which enlarges the view of an editor's object enabling you to see more detail. You can continuous zoomIn and zoomOut either using zoom tools or keyboard.

You can assign Increases and decreases of zooming range using [zoomStep](#) property

## HTML

```
<textarea id="editor"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
toolsList: ["view"],
tools: { view:["zoomIn","zoomOut"]},
zoomStep: 0.1
});
});
}
```



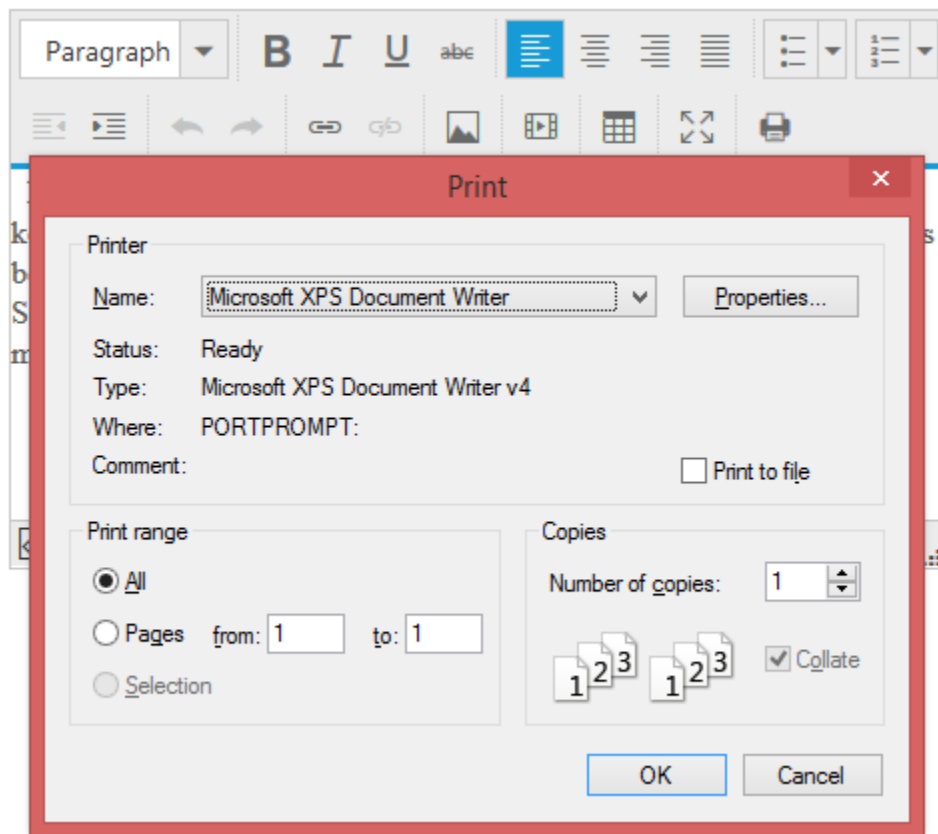
## Print

The editor provides print tools which use to print the contents of the editor.



## HTML

```
<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nser t table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
toolsList: ["print"],
tools: { print:["print"]}
});
});
}
```



## Working with Lists

The editor provides tools to makes your content as list such as an ordered and unordered list.

### Create a Lists

By default, [Insert Lists](#) tool is enabled in the editor's toolbar.The editor's have ordered and unordered list types.

## HTML

```

<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nsert table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
toolsList: ["lists"],
tools: {
lists: ["unorderedList", "orderedList"]
}
});
});
}

```

### Custom Lists

You can use [custom lists](#) tools to insert lists with custom behaviors. You can create a list with related attributes (such as listImage, listStyle, title, name, and text) using the custom list tool. Ordered and Unordered list having own customize ways to insert a list into the editor's content.

- [Insert a customOrderedList](#)
- [Insert a customUnorderedList](#)

#### *Insert a customOrderedList*

you need to enable [customOrderedList](#) tool on the editor's toolbar.

The customOrderedList having below options for an ordered list customization.

| Option                    | Summary  |
|---------------------------|--|
| <a href="#">name</a>      | Specifies the name for customOrderedList item.       |
| <a href="#">tooltip</a>   | Specifies the title for customOrderedList item.      |
| <a href="#">css</a>       | Specifies the styles for customOrderedList item.     |
| <a href="#">text</a>      | Specifies the text for customOrderedList item.       |
| <a href="#">listStyle</a> | Specifies the list style for customOrderedList item. |
| <a href="#">listImage</a> | Specifies the image for customOrderedList item.      |

### HTML

```

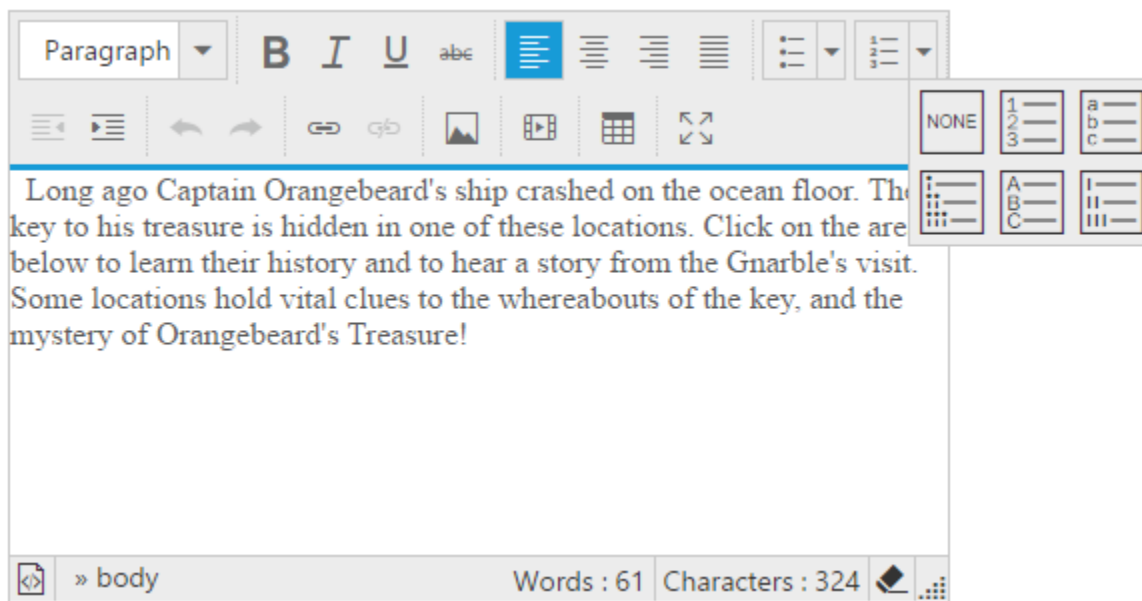
<textarea id="editor"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {

```

```
var sample = new ej.RTE($("#rteSample"), {
  value:
    "The RichTextEditor (RTE) control enables you to edit the contents with i  

    nsert table and images," +
    " it also provides a toolbar that helps to apply rich text formats to the  

    content entered in the TextArea.",
  toolsList: ["lists"],
  tools: {
    lists: ["orderedList"],
    customOrderedList: [{
      name: "orderedInsert",
      tooltip: "Custom OrderedList",
      css: "e-rte-toolbar-icon e-rte-listitems customOrdered",
      text: "Lower-Greek",
      listStyle:"lower-greek"
    }]
  }
});
```



*Insert a customUnorderedList*

you need to enable [customUnorderedList](#) tool on the editor's toolbar.

The customUnorderedList having below options for an unordered list customization.

| Option                  | Summary  |
|-------------------------|--|
| <a href="#">name</a>    | Specifies the name for customUnorderedList item.   |
| <a href="#">tooltip</a> | Specifies the title for customUnorderedList item.  |
| <a href="#">css</a>     | Specifies the styles for customUnorderedList item. |
| <a href="#">text</a>    | Specifies the text for customUnorderedList item.   |

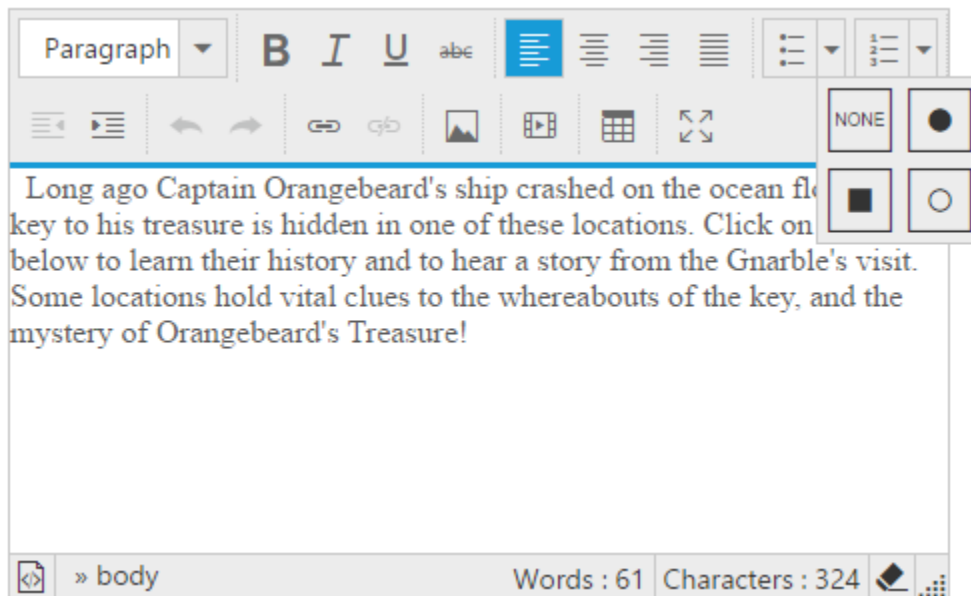
|                           |  |
|---------------------------|--|
| <a href="#">listStyle</a> | Specifies the list style for customUnorderedList item. |
| <a href="#">listImage</a> | Specifies the image for customUnorderedList item.      |

**HTML**

```

<textarea id="editor"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nser table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
toolsList: ["lists"],
tools: {
lists: ["unorderedList"],
customUnorderedList: [{
name: "UnorderedInsert",
tooltip: "Custom UnorderedList",
css: "e-rte-toolbar-icon e-rte-unlistitems customUnOrdered",
text: "Smiley",
listImage:"url('../content/images/rte/Smiley-GIF.gif') "
}]
}
});
});
}

```



## Clean unwanted elements and styles when copy paste from Microsoft Word

While copy pasting content from MSWord document, the content will be processed in the paste action event. [pasteCleanupSettings](#) API can be used for removing unwanted elements.

This will convert an unformatted HTML element (MOS XML format) content to a proper HTML element. Table elements also will be converted with proper elements.

| Options                        | Description  |
|--------------------------------|--|
| <a href="#">listConversion</a> | List Conversion option convert the list elements into a proper format pasted from MSWord document to editor. |
| <a href="#">cleanCSS</a>       | Clean CSS is used to clean the unwanted css in the elements pasted from MSWord document to editor            |
| <a href="#">removeStyles</a>   | Remove styles will remove all styles in the elements pasted from MSWord document to editor.                  |
| <a href="#">cleanElements</a>  | Clean Elements is used to clean the unwanted elements pasted from MSWord document to editor.                 |

## HTML

```
<textarea id="rteSample">
<p><b>Description:</b></p>
<p>The Rich Text Editor (RTE) control is easy to render in the
client side. Customers can easily edit the contents and get the HTML
content for
the displayed content. A rich text editor control provides users with a
toolbar
that helps them to apply rich text formats to the text entered in the
text
area. </p></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
pasteCleanupSettings:{
listConversion:true,
cleanCSS:true,
removeStyles:true,
cleanElements:true
}
});
});
}
```

Pasted Content before Cleanup:

**Key Features:**

Retrieve data from a remote endpoint.

Synchronize updates—create, update, delete to and from a remote endpoint.

Calculate and maintain aggregates, sorting order and paging.

Provide a query mechanism via filter expressions.

| Name      | Type   | Description              |
|-----------|--------|--------------------------|
| data      | Object | JSON data or JSON array  |
| tableName | String | name of the source table |

» body » p » span

Words : 63 Characters : 555

```

<body spellcheck="false" autocorrect="off" contenteditable="true">
  <h2>...</h2>
  <p class="MsoListParagraphCxSpFirst" style="text-indent:-.25in;mso-list:l0 level1 lfo1">...</p>
  <p class="MsoListParagraphCxSpMiddle" style="text-indent:-.25in;mso-list:l0 level1 lfo1">...</p>
  <p class="MsoListParagraphCxSpMiddle" style="text-indent:-.25in;mso-list:l0 level1 lfo1">...</p>
  <p class="MsoListParagraphCxSpLast" style="text-indent:-.25in;mso-list:l0 level1 lfo1">...</p>
  <table class="MsoTableGrid e-rte-table" border="1" cellpadding="0" cellspacing="1" width="678.6666870117188"
    style="border: 1px solid rgb(92, 92, 92); border-collapse: separate; border-spacing: 2px;">...</table>
  <p></p>
  <p class="MsoNormal">...</p>
</body>

```

Pasted Content after Cleanup:

**Key Features:**

- Retrieve data from a remote endpoint.
- Synchronize updates—create, update, delete to and from a remote endpoint.
- Calculate and maintain aggregates, sorting order and paging.
- Provide a query mechanism via filter expressions.

| Name      | Type     | Description                                 |
|-----------|----------|---|
| data      | Object   | JSON data or JSON array                     |
| tableName | String   | name of the source table                    |
| query     | ej.Query | Sets the default query for the data source. |

» body » p » span

Words : 52 Characters : 386

```
<body spellcheck="false" autocorrect="off" contenteditable="true">
  <p>...</p>
  <h2>Key Features:</h2>
  <ul level="1">...</ul>
  <table border="1" cellpadding="0" class="e-rte-table" cellspacing="1" width="672.8240793359375" style="border: 1px solid rgb(92, 92, 92); border-collapse: separate; border-spacing: 2px;">...</table>
  <p>...</p>
  <p>...</p>
  <ol>...</ol>
</body>
```

## Localization

The editor provides option to localize its strings, it is used to adapting the editor to a particular local language. By default, the editor will use the US English (en-US) as its language. Please find the table with list of keys and their corresponding values for default language (en-US).

### JAVASCRIPT

```
ej.RTE.Locale["en-US"] = {
  bold: "Bold",
  italic: "Italic",
  underline: "Underline",
  strikethrough: "Strikethrough",
  superscript: "Superscript",
  subscript: "Subscript",
  justifyCenter: "Align text center",
  justifyLeft: "Align text left",
  justifyRight: "Align text right",
  justifyFull: "Justify",
  unorderedList: "Insert unordered list",
```

```
orderedList: "Insert ordered list",
indent: "Increase Indent",
fileBrowser: "File Browser",
outdent: "Decrease Indent",
cut: "Cut",
copy: "Copy",
paste: "Paste",
paragraph: "Paragraph",
undo: "Undo",
redo: "Redo",
upperCase: "Upper Case",
lowerCase: "Lower Case",
clearAll: "Clear All",
clearFormat: "Clear Format",
createLink: "Insert/Edit Hyperlink",
removeLink: "Remove Hyperlink",
tableProperties: "Table Properties",
insertTable: "Insert",
deleteTables: "Delete",
imageProperties: "Image Properties",
openLink: "Open Hyperlink",
image: "Insert image",
video: "Insert video",
editTable: "Edit Table Properties",
embedVideo: "Paste your embed code below",
viewHtml: "View HTML",
fontName: "Select font family",
fontSize: "Select font size",
fontColor: "Select color",
format: "Format",
backgroundColor: "Background color",
style: "Styles",
deleteAlert: "Are you sure you want to clear all the contents?",
copyAlert: "Your browser doesn't support direct access to the clipboard.
Please use the Ctrl+C keyboard shortcut instead of copy operation.",
pasteAlert: "Your browser doesn't support direct access to the clipboard.
Please use the Ctrl+V keyboard shortcut instead of paste operation.",
cutAlert: "Your browser doesn't support direct access to the clipboard.
Please use the Ctrl+X keyboard shortcut instead of cut operation.",
videoError: "The text area can not be empty",
imageWebUrl: "Web Address",
imageAltText: "Alternate text",
dimensions: "Dimensions",
constrainProportions: "Constrain Proportions",
linkWebUrl: "Web Address",
imageLink: "Image as Link",
imageBorder: "Image Border",
imageStyle: "Style",
linkText: "Text",
linkToolTip: "Tooltip",
html5Support: "This tool icon only enabled in HTML5 supported browsers",
linkOpenInNewWindow: "Open link in new window",
tableColumns: "No.of Columns",
tableRows: "No.of Rows",
tableWidth: "Width",
tableHeight: "Height",
tableCellSpacing: "Cell spacing",
```



```
tableCellPadding: "Cell padding",
tableBorder: "Border",
tableCaption: "Caption",
tableAlignment: "Alignment",
textAlign: "Text align",
dialogUpdate: "Update",
dialogInsert: "Insert",
dialogCancel: "Cancel",
dialogApply: "Apply",
dialogOk: "Ok",
createTable: "Insert Table",
insertTable: "Insert",
addColumnLeft: "Insert Columns to the Left",
addColumnRight: "Insert Columns to the Right",
addRowAbove: "Insert Rows Above",
addRowBelow: "Insert Rows Below",
deleteRow: "Delete entire row",
deleteColumn: "Delete entire column",
deleteTable: "Delete Table",
customTable: "Create custom table...",
characters: "Characters",
words: "Words",
general: "General",
advanced: "Advanced",
table: "Table",
row: "Row",
column: "Column",
cell: "Cell",
solid: "Solid",
dotted: "Dotted",
dashed: "Dashed",
doubled: "Double",
maximize: "Maximize",
resize: "Minimize",
swatches: "Swatches",
paragraph: "Paragraph",
quotation: "Quotation",
heading1: "Heading 1",
heading2: "Heading 2",
heading3: "Heading 3",
heading4: "Heading 4",
heading5: "Heading 5",
heading6: "Heading 6",
segoeui: "Segoe UI",
arial: "Arial",
couriernew: "Courier New",
georgia: "Georgia",
impact: "Impact",
lucidaconsole: "Lucida Console",
tahoma: "Tahoma",
timesnewroman: "Times New Roman",
trebuchetms: "Trebuchet MS",
verdana: "Verdana",
disc: "Disc",
circle: "Circle",
square: "Square",
number: "Number",
```

```

loweralpha:"Lower Alpha",
upperalpha:"Upper Alpha",
lowerroman:"Lower Roman",
upperroman:"Upper Roman",
none:"None",
linktooltip:"ctrl + click to follow link",
charSpace: "Characters (with spaces)",
charNoSpace: "Characters (no spaces)",
wordCount: "Word Count",
left:"Left",
right:"Right",
center:"Center",
zoomIn: "Zoom In",
zoomOut: "Zoom Out",
print: "Print",
FindAndReplace:"Find and Replace",
Find:"Find",
MatchCase:"Match Case",
WholeWord:"Whole Word",
ReplaceWith:"Replace with",
Replace:"Replace",
ReplaceAll:"Replace All",
FindErrorMsg:"Couldn't find the specified word."
};
var format = [
{ text: "Paragraph", value: "p", spriteCssClass: "e-paragraph" },
{ text: "Quotation", value: "blockquote", spriteCssClass: "e-quotation"
},
{ text: "Heading 1", value: "h1", spriteCssClass: "e-h1" },
{ text: "Heading 2", value: "h2", spriteCssClass: "e-h2" },
{ text: "Heading 3", value: "h3", spriteCssClass: "e-h3" },
{ text: "Heading 4", value: "h4", spriteCssClass: "e-h4" },
{ text: "Heading 5", value: "h5", spriteCssClass: "e-h5" },
{ text: "Heading 6", value: "h6", spriteCssClass: "e-h6" }
];

```

---

**Note:** The culture name has to be specified in a standard format such as [Language Code]-[County/Region Code].

---

To localize the editor's strings with your own localization, copy the default language informations and localize the strings in the values column. For example, to localize the editor in German language ("de-DE").

#### JAVASCRIPT

```

ej.RTE.Locale["de-DE"] = {
bold: "fett",
italic: "kursiv",
underline: "unterstreichen",
strikethrough: "Durchgestrichen",
superscript: "Überschrift",
subscript: "Index",
justifyCenter: "Text-Zentrum",
justifyLeft: "Ausrichten von Text links",
justifyRight: "Ausrichten von Text rechts",
justifyFull: "rechtfertigen",

```

```
fileBrowser: "Datei-Browser",
unorderedList: "Legen Sie ungeordnete Liste",
orderedList: "Geordnete Liste einfügen",
indent: "Einzug",
outdent: "Einzug verkleinern",
cut: "schneiden",
copy: "Kopieren",
paste: "Paste",
undo: "lösen",
redo: "Wiederherstellen",
upperCase: "Großbuchstaben",
lowerCase: "Kleinbuchstaben",
clearAll: "Alles",
clearFormat: "Klar Format",
createLink: "Einfügen / Hyperlink Bearbeiten",
removeLink: "fjern Hyperlink",
tableProperties: "Tabelleneigenschaften",
insertTable: "Einfügen",
deleteTables: "Löschen",
imageProperties: "Bildeigenschaften",
openLink: "Verbindung öffnen",
image: "Bild einfügen",
video: "Legen Video",
embedVideo: "Fügen Sie den Embed-Code unten",
viewHtml: "Blick HTML",
fontName: "Wählen Sie Schriftfamilie",
fontSize: "Wählen Sie Schriftgröße",
fontColor: "Wählen Sie die Farbe",
format: "Format",
backgroundColor: "Hintergrundfarbe",
style: "Styles",
deleteAlert: "Sind Sie sicher, dass Sie alle Inhalte löschen?",
copyAlert: "Ihr Browser unterstützt leider keinen direkten Zugriff auf  
die Zwischenablage. Bitte verwenden Sie die Ctrl + C -Tastatur statt  
Kopiervorgang Verknüpfung .",
pasteAlert: "Ihr Browser unterstützt leider keinen direkten Zugriff auf  
die Zwischenablage. Bitte verwenden Sie die Ctrl + V Tastenkombination  
statt Einfügen.",
cutAlert: "Ihr Browser unterstützt leider keinen direkten Zugriff auf die  
Zwischenablage. Bitte verwenden Sie die Ctrl + X Tastenkombination statt  
Schneidevorgang.",
videoError: "Der Textbereich darf nicht leer sein",
imageWebUrl: "Webadresse",
imageAltText: "Bildbeschreibung",
dimensions: "Größe",
constrainProportions: "Proportionen",
linkWebUrl: "Webadresse",
linkText: "Text",
linkToolTip: "Tooltip",
html5Support: "Dieses Werkzeug-Symbol nur in HTML5 aktiviert  
unterstützten Browser",
linkOpenInNewWindow: "Link in einem neuen Fenster",
tableColumns: "Spalten",
tableRows: "Zeilen",
solid: "solide",
dashed: "gestippelte",
dotted: "stippel",
```

```
doubled: "dubbele",
buttonApply: "toepassen",
buttonCancel: "annuleren",
closeIcon: "dicht",
tableWidth: "Tischbreite",
tableHeight: "Tischhöhe",
tableCellSpacing: "Zellenabstand",
tableCellPadding: "Zellauffüllung",
tableBorder: "Grenze",
tableCaption: "Beschriftung",
tableAlignment: "Ausrichtung",
dialogUpdate: "Aktualisierung",
dialogInsert: "einfügen",
dialogCancel: "stornieren",
dialogOk: "Ok",
createTable: "Tabelle erstellen",
addColumnLeft: "Spalte links hinzufügen",
addColumnRight: "In Spalte auf der rechten",
addRowAbove: "Zeile hinzufügen oben",
addRowBelow: "Zeile hinzufügen unten",
deleteRow: "Löschen Sie die Zeile",
deleteColumn: "Löschen Sie die Spalte",
deleteTable: "Löschen Sie die Tabelle",
insertTable: "Einfügen",
customTable: "von benutzerdefinierten Tabelle",
characters: "Charaktere",
dialogApply: "anwenden",
textAlign: "Text ausrichten",
imageLink: "Bild als Link zu",
imageBorder: "Bild Rand",
imageStyle: "Stil",
editTable: "Tabelle Eigenschaften bearbeiten",
words: "Wörter",
general: "allgemein",
advanced: "fortgeschritten",
table: "Tisch",
row: "Reihe",
column: "Spalte",
cell: "Zelle",
maximize: "Maximieren",
resize: "Minimieren",
swatches: "Farbfelder",
paragraph: "Absatz",
quotation: "Zitat",
heading1: "Kopf 1",
heading2: "Kopf 2",
heading3: "Kopf 3",
heading4: "Kopf 4",
heading5: "Kopf 5",
heading6: "Kopf 6",
disc: "Scheibe",
circle: "Kreis",
square: "Platz",
number: "Anzahl",
loweralpha: "Lower Alpha",
upperalpha: "Ober Alpha",
lowerroman: "Lower Roman",
```

```

upperroman:"Ober Roman",
none:"Keine",
linktooltip:"ctrl + Klick Link zu folgen",
charSpace: "Zeichen (mit Leerzeichen )",
charNoSpace: "Zeichen (ohne Leerzeichen)",
wordCount: "Wortzahl",
right:"Recht",
left:"links",
center:"Center",
FindAndReplace:"Suchen und Ersetzen",
Find:"Finden",
MatchCase:"Kleinschreibung",
WholeWord:"Ganze Welt",
ReplaceWith:"Ersetzen mit",
Replace:"Ersetzen",
ReplaceAll:"Alles ersetzen",
FindErrorMsg:"Konnte den angegebenen Wort gefunden ."
};
var format_DE = [
{ text: "Absatz", value: "p", spriteCssClass: "e-paragraph" },
{ text: "Zitat", value: "blockquote", spriteCssClass: "e-quotation" },
{ text: "Kopf 1", value: "h1", spriteCssClass: "e-h1" },
{ text: "Kopf 2", value: "h2", spriteCssClass: "e-h2" },
{ text: "Kopf 3", value: "h3", spriteCssClass: "e-h3" },
{ text: "Kopf 4", value: "h4", spriteCssClass: "e-h4" },
{ text: "Kopf 5", value: "h5", spriteCssClass: "e-h5" },
{ text: "Kopf 6", value: "h6", spriteCssClass: "e-h6" }
];

```

You can set the [locale](#) property of the editor to the new language.

### HTML

```

<textarea id="rteSample"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"Das RichTextEditor (RTE) Steuerung ermöglicht Ihnen, den Inhalt mit Eins
atz Tisch und Bilder zu bearbeiten ," +
" sondern bietet auch eine Werkzeugleiste , die Rich-Text- Format ," +
" um die in der Textarea eingegeben Gehalt gelten können.",
locale: "de-DE", format: format_DE
});
});
}

```

### Keyboard Support

The editor has full keyboard accessibility that includes shortcuts to open and other actions with toolbar items, drop-down lists, and dialogs.

| Press | To do this |
|-------|------------|
|-------|------------|

|                           |   |
|---------------------------|---|
| Arrow keys                | When a toolbar is focused Move between options in an open drop-down list      |
| Enter                     | Execute the selected button, drop-down item                                   |
| ESC                       | Cancel an action or close the dialog  |
| HOME<br>END               | Go to first/last of the line  |
| CTRL + HOME<br>CTRL + END | Go to the beginning/end of a content  |
| PAGE UP<br>PAGE DOWN      | Scroll up/down page in the content  |
| CTRL + A                  | Select all content  |
| CTRL + C                  | Copy the selected text or image   |
| CTRL + X                  | Cut the selected text   |
| CTRL + V                  | Paste text or image   |
| CTRL + Z                  | Undo the last action  |
| CTRL + Y                  | Redo the last action  |
| CTRL + U                  | Applies underline format with the selected content.                           |
| CTRL + I                  | Applies italic format with the selected content.                              |
| CTRL + B                  | Applies bold format with the selected content.                                |
| CTL+SHIFT+S               | Applies strike format with the selected content.                              |
| CTRL+K                    | Opens hyperlink dialog for attaching the hyperlink with the selected content. |
| CTRL+SHIFT+K              | Removes hyperlink from the selected content.                                  |
| CTRL+E                    | Justify center.   |
| CTRL+J                    | Justify Full.   |

|                  |   |
|------------------|---|
| CTRL+L           | Justify Left.   |
| CTRL+R           | Justify Right.  |
| CTRL+M           | Increases the indent of the focused content.                              |
| CTRL+SHIFT+M     | Decreases the indent of the focused content.                              |
| CTRL +SHIFT+H    | Opens the HTML source of the editor content.                              |
| CTRL+ SHIFT +I   | Inserts or edit image with editor content.                                |
| CTRL+SHIFT+V     | Inserts video to the content.   |
| CTRL+ SHIFT +F   | Expands the editor area to full screen mode.                              |
| CTRL+SHIFT+R     | Clear text formats from the selected content.                             |
| CTRL+SHIFT+C     | Opens custom table dialog, to insert a customized table with the content. |
| CTRL+SHIFT+E     | Opens edit table dialog, for editing the table property.                  |
| CTRL+SHIFT+LEFT  | Inserts a column before the focused cell.                                 |
| CTRL+SHIFT+RIGHT | Inserts a column after the focused cell.                                  |
| CTRL+SHIFT+UP    | Inserts a row above the focused cell.                                     |
| CTRL+SHIFT+DOWN  | Inserts a row below the focused cell.                                     |
| CTRL+ALT+C       | Removes focused column from the table.                                    |
| CTRL+ALT+R       | Removes focused row from the table.                                       |
| CTRL+ALT+A       | Deletes entire Table  |
| CTRL+SHIFT+<     | Decrease font size for the selected content.                              |
| CTRL+SHIFT+>     | Increase font size for the selected content.                              |
| CTRL+SHIFT+L     | Replaces the selected content to lowercase content.                       |
| CTRL+SHIFT+U     | Replaces the selected content to uppercase content.                       |
| CTRL+SHIFT +=    | Enables the superscript with the selected content.                        |

|              |   |
|--------------|---|
| CTRL +=      | Enables the subscript with the selected content.              |
| CTRL+SHIFT+O | Inserts the ordered list element with the selected content.   |
| CTRL+ALT+O   | Inserts the unordered list element with the selected content. |
| CTRL+F10     | Clears the entire editor content.                             |

To disable the keyboard navigation, set the [allowKeyboardNavigation](#) property of the editor to false (its default value is true). It will disable all the keyboard navigation shortcuts except for the UP/DOWN keys and PAGE UP/PAGE DOWN keys.

### HTML

```
<textarea id="editor"></textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
value:
"The RichTextEditor (RTE) control enables you to edit the contents with i
nser table and images," +
" it also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.",
allowKeyboardNavigation: false
});
});
}
```

**Note:** In order to make the tab key for form element navigation –disable the **enableTabKeyNavigation** property.

### How To

#### Add Google web fonts to editor

To use web fonts in EJ RTE, it is not needed for the web fonts to be present in local machine. To add the web fonts to EJ RTE, we need to append the web font link to the RTE Iframe <head> tag. We can achieve this in the “create” event of the RTE as shown below.

### HTML

```
<textarea id="rteSample" rows="10" cols="30" style="width: 740px; height:
440px">
Description:
The Rich Text Editor (RTE) control is an easy to render in
client side.Customer easy to edit the contents and get the HTML content
for
the displayed content.A rich text editor control provides users with a
toolbar
that helps them to apply rich text formats to the text entered in the
text
area.
```



```

</textarea>
var fonts = [
{ text: "Segoe UI", value: "Segoe UI" },
//Added from Google web fonts
{ text: "Noto Sans", value: "Noto Sans" },
{ text: "Roboto", value: "Roboto" },
{ text: "Great vibes", value: "Great Vibes,cursive" }
];
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"),{
width: "550px",
showFooter: true,
fontName:fonts,
tools: {
font: ["fontName", "fontSize", "fontColor", "backgroundColor"],
style: ["bold", "italic", "underline", "strikethrough"],
alignment: ["justifyLeft", "justifyCenter", "justifyRight",
"justifyFull"],
lists: ["unorderedList", "orderedList"],
copyPaste: ["cut", "copy", "paste"],
doAction: ["undo", "redo"],
clear: ["clearFormat", "clearAll"]
},
create: function () {
var $head = this._rteIframe.contents().find("head");
var url = "Styles/styles.css";
//Added two Google web fonts ("Roboto" and "Great vibes")
$head.append($("#<link/>", { rel: "stylesheet", href:
'http://fonts.googleapis.com/css?family=Roboto', type: "text/css" }));
$head.append($("#<link/>", { rel: "stylesheet", href:
'http://fonts.googleapis.com/css?family=Great+Vibes', type: "text/css"
}));
}
});
});
}

```

In the above sample, you can see that we have added two Google web fonts ("Roboto" and "Great vibes") to ejRTE. [Demo Link](#)

#### Increase RTE max word count

To increase the RTE content maximum word count, we suggest you to set the [maxLength](#) property for it. By default, maxLength value is 7000, assign it with a value based on your requirement.

Refer the following API reference link: [Link](#)

#### Add multiple editor instances to a single page

This is in fact a common use case, especially when you may wish to break your content into sections (e.g. titles, paragraphs) that the user can edit individually.

*Multiple editor instances sharing same RTE properties*

In the following example, the page is broken into two separate editable areas, each sharing the same RTE configuration. Each individual `textarea` is provided the same class of 'myEdit'.

**HTML**

```
<textarea class="myEdit" rows="10" cols="30" style="width: 740px; height: 440px">
Title
</textarea>
<br>
<textarea class="myEdit" rows="10" cols="30" style="width: 740px; height: 440px">
Description
</textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($(".myEdit"), {
width: "100%",
minWidth: "100px",
isResponsive: true,
showToolbar: false,
height: "50px"
});
});
}
```

*Multiple editor instances sharing a unique RTE properties*

In this next example each editable area will be loaded with an instance of RTE with a unique configuration. This is especially helpful when different content areas have different needs. For example, you may want to provide a very simple configuration for editing titles and a more complete configuration for editing body content. This is accomplished by defining a RTE property for each desired configuration.

**HTML**

```
<textarea class="myEdit1" rows="10" cols="30" style="width: 740px; height: 440px">
Title
</textarea>
<br>
<textarea class="myEdit2" rows="10" cols="30" style="width: 740px; height: 440px">
Description
</textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($(".myEdit1"), {
width: "100%",
minWidth: "100px",
isResponsive: true,
showToolbar: false,
```

```
height: "50px"
});
var sample1 = new ej.RTE($("#myEdit2"), {
width: "100%",
minWidth: "100px",
isResponsive: true
});
});
}
```

### Set the horizontal scroller rather than text wrapping in the RTE?

This can be achieved by setting the CSS “whitespace” as nowrap in RTE body element in the create event of RTE as shown below code:

#### HTML

```
<textarea id="rteSample" rows="10" cols="30" style="width: 740px; height: 440px">
The RichTextEditor (RTE) control enables you to edit the contents with
insert table and images
It also provides a toolbar that helps to apply rich text formats to the
content entered in the TextArea.
</textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
width: "100%",
minWidth: "100px",
create: "onCreate"
});
});
}
function onCreate() {
//Access the content of the RTE within IFrame and set the style
$('#rteSample_iframe').contents().find('body').attr("style", "white-
space: nowrap");
}
```

### Set Toolbar Height

We do not have any property for adjusting the height of the RTE Toolbar. But it is possible to adjust the height by overriding the class of the RTE toolbar. Override the class as below,

#### CSS

```
<style>
.e-rte .e-js.e-toolbar{
height: 100px;
}
</style>
```

### Add Separator in the Toolbar

we can add separator in the RTE toolbar list. We have a property [“enableSeparator”](#) in the toolbar control. So we need to set this property as true by creating the object of toolbar in the “preRender” event of RTE as shown below code:

#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
enableSeparator: true
});
}
}

```

### Custom image for the Tools

This requirement can have achieved by using [“cssClass”](#) API of RichTextEditor component. It is used to customize the default “CSS” styles of the control. Using this API to define our own custom “CSS” and images to overwrite the default “CSS” styles of the control.

#### HTML

```

<textarea id="rteSample" rows="10" cols="30" style="width: 740px; height: 440px">
The RichTextEditor (RTE) control enables you to edit the contents with insert table and images
It also provides a toolbar that helps to apply rich text formats to the content entered in the TextArea.
</textarea>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RTEComponent {
$(function () {
var sample = new ej.RTE($("#rteSample"), {
cssClass: "dark"
});
});
}

```

Apply the following style and In the below sample, the sprite image has been used for the icons

#### HTML

```

<style type="text/css">
.dark .e-rte-toolbar-icon {
background: url(themes/icons/ui-icons-default.png);
}
.dark .e-active .e-rte-toolbar-icon, .dark .e-rte-toolbar-icon:hover {
background: url(themes/icons/ui-icons-active.png);
}
.e-rte-toolbar-icon.bold:before, .e-rte-toolbar-icon.italic:before, .e-rte-toolbar-icon.underline:before, .e-rte-toolbar-icon.strikethrough:before, .e-rte-toolbar-icon.justifyLeft:before, .e-

```

```
rte-toolbar-icon.justifyCenter:before, .e-rte-toolbar-
icon.justifyRight:before, .e-rte-toolbar-icon.justifyFull:before, .e-rte-
toolbar-icon.unorderedList:before, .e-rte-toolbar-
icon.orderedList:before, .e-rte-toolbar-icon.undo:before, .e-rte-toolbar-
icon.redo:before, .e-rte-toolbar-icon.createLink:before, .e-rte-toolbar-
icon.image:before, .e-rte-toolbar-icon.video:before, .e-rte-toolbar-
icon.createTable:before {
content: none;
}
.dark .e-rte-toolbar-icon.outdent, .dark .e-rte-toolbar-icon.indent {
background: none;
}
.dark .e-rte-toolbar-icon.bold, .dark .e-active .e-rte-toolbar-icon.bold
{
background-position: -288px 49px;
}
.dark .e-rte-toolbar-icon.italic, .dark .e-active .e-rte-toolbar-
icon.italic {
background-position: -313px 49px;
}
.dark .e-rte-toolbar-icon.underline, .dark .e-active .e-rte-toolbar-
icon.underline {
background-position: -338px 49px;
}
.dark .e-rte-toolbar-icon.strikethrough, .dark .e-active .e-rte-toolbar-
icon.strikethrough {
background-position: -364px 49px;
}
.dark .e-rte-toolbar-icon.justifyLeft, .dark .e-active .e-rte-toolbar-
icon.justifyLeft {
background-position: -389px 112px;
}
.dark .e-rte-toolbar-icon.justifyCenter, .dark .e-active .e-rte-toolbar-
icon.justifyCenter {
background-position: -338px 112px;
}
.dark .e-rte-toolbar-icon.justifyRight, .dark .e-active .e-rte-toolbar-
icon.justifyRight {
background-position: -416px 112px;
}
.dark .e-rte-toolbar-icon.justifyFull, .dark .e-active .e-rte-toolbar-
icon.justifyFull {
background-position: -364px 112px;
}
.dark .e-rte-toolbar-icon.unorderedList, .dark .e-active .e-rte-toolbar-
icon.unorderedList {
background-position: -131px 89px;
}
.dark .e-rte-toolbar-icon.orderedList, .dark .e-active .e-rte-toolbar-
icon.orderedList {
background-position: -104px 89px;
}
.dark .e-rte-toolbar-icon.undo, .dark .e-active .e-rte-toolbar-icon.undo
{
background-position: -262px 49px;
}
```

```
.dark .e-rte-toolbar-icon.redo, .dark .e-active .e-rte-toolbar-icon.redo
{
background-position: -234px 49px;
}
.dark .e-rte-toolbar-icon.createLink, .dark .e-active .e-rte-toolbar-
icon.createLink {
background-position: -470px 69px;
}
.dark .e-rte-toolbar-icon.image, .dark .e-active .e-rte-toolbar-
icon.image {
background-position: -287px 113px;
}
.dark .e-rte-toolbar-icon.video, .dark .e-active .e-rte-toolbar-
icon.video {
background-position: -311px 70px;
}
.dark .e-rte-toolbar-icon.createTable, .dark .e-active .e-rte-toolbar-
icon.createTable {
background-position: -1px 27px;
}
</style>
```

## Rotator

### Overview

The Typescript Rotator control displays a set of sliding images or images and content, or content with user-defined transition between them. It supports Data Binding, Thumbnail, Pager, dynamic number of slide move options and all custom animations. It supports all types of image formats (JPEG, GIF and so on).

### Key Features

**Data Binding:** Supports data binding with local and remote data.

**Image with Content:** Supports to render an image, content, or image with content.

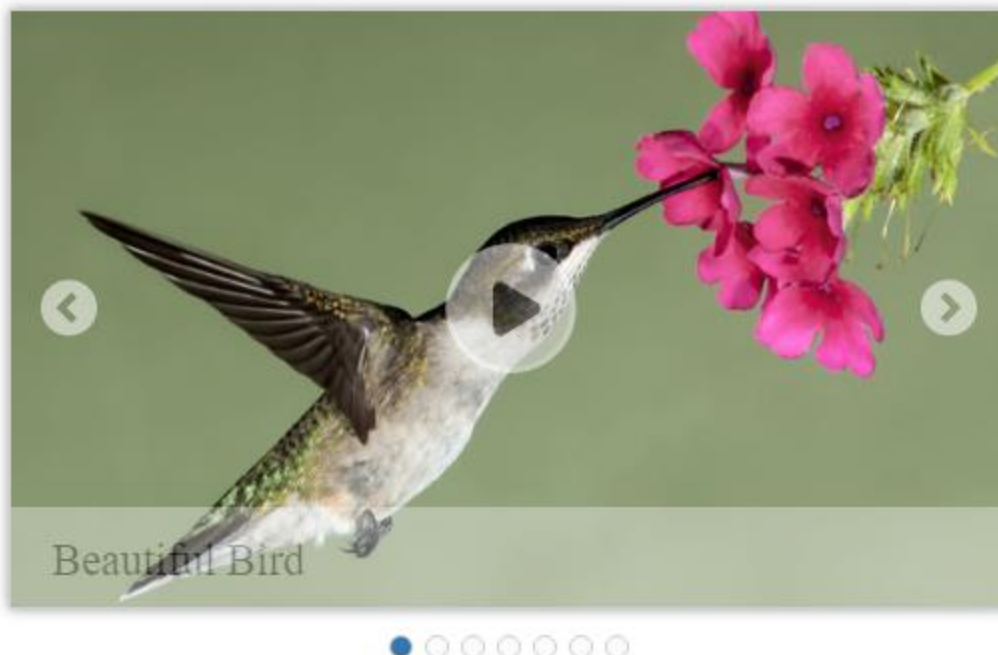
**Customized slides:** Supports dynamic (one or more) number of slides and to move dynamic (one or more) number of slides at a time.

**Auto-play:** Supports auto-play mode for slide transition.

**Pager and Thumbnail:** Support to navigate between slides with thumbnail images or paging

### Getting Started

This section helps to get started of the Rotator control in a typescript application.



### Create a Rotator

The following step by step guidelines will help you to add a Rotator control.

- 1) Refer the common Typescript [Getting Started Documentation]() to create an application and add necessary scripts and styles for rendering the Rotator control.
- 2) Create an HTML page and add the scripts and css references in the order mentioned in the following code example.

### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Typescript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/fl
at-azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.
web.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add Rotator code here-->
</body>
</html>
```

- 3) Add the below mentioned code with in the tag to render the Rotator control in the application.

### HTML

```
<div class="frame">
<ul id="sliderContent">
```

```
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
</ul>
</div>
```

### Initialize the control

Initialize the Rotator in ts file by using the ej.Rotator method.

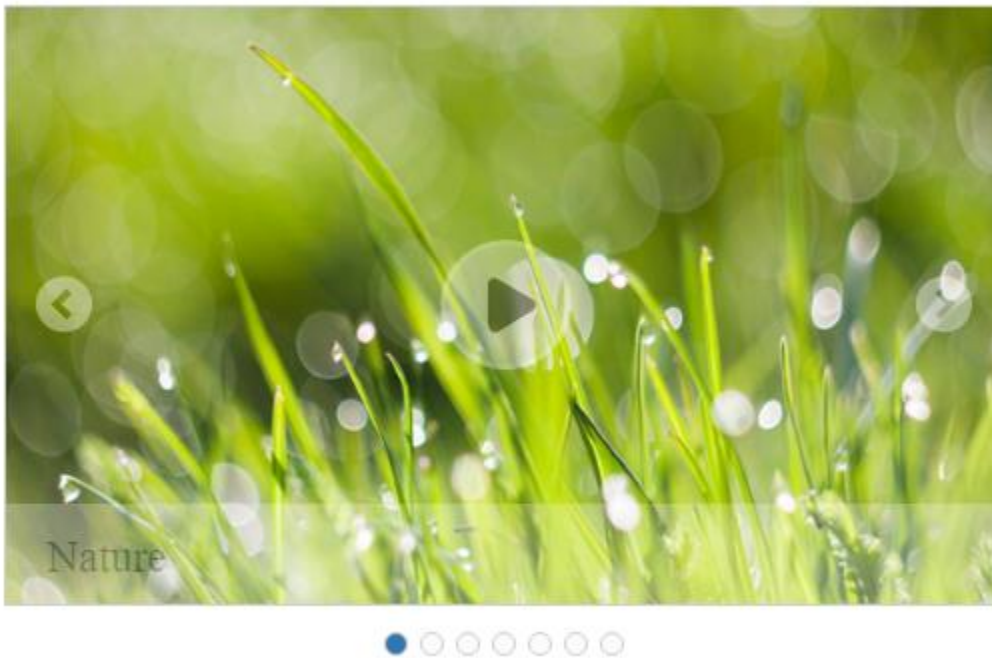
### TS

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
$(function () {
var rotatorInstance = new ej.Rotator($("#sliderContent"), {
slideWidth: "100%",
frameSpace: "0px",
slideHeight: "auto",
displayItemsCount: "1",
navigateSteps: "1",
pagerPosition: "outside",
orientation: "horizontal",
showPager: true,
```



```
enabled: true,  
showCaption: true,  
allowKeyboardNavigation: true,  
showPlayButton: true,  
isResponsive: true,  
animationType: "slide",  
});  
});  
}
```

Get the following output from the above mentioned code



---

*Note: You can find the Rotator properties from the [API reference](#) document*

---

### Data Binding

**Rotator** provides a flexible approach for binding the data from various data sources. There are various properties in **Toolbar** for **Data Binding**. The value set to this property is **object** type.

### Data fields and Configuration

The following sub-properties provide a way to bind the data either locally/remotely to the **Rotator** control. The value set to this property is **object** type.

### DataSource

This property specifies the list of data that contains a set of data fields. Each data value is used to render an item for the **Rotator**. The value set to this property is **object** type.

### Fields

It defines mapping fields for the data items of the **Rotator**. The value set to this property is **object** type.

### Text

It specifies the text content of the tag. The value set to this property is **string** type.

## URL

This property specifies the **URL** for an image. The value set to this property is **string** type.

## Query

This property retrieves data from remote data. This property is applicable only when a remote data source is used. Each data value is used to render an item for the **Rotator**. The value set to this property is **object** type.

## Local data binding

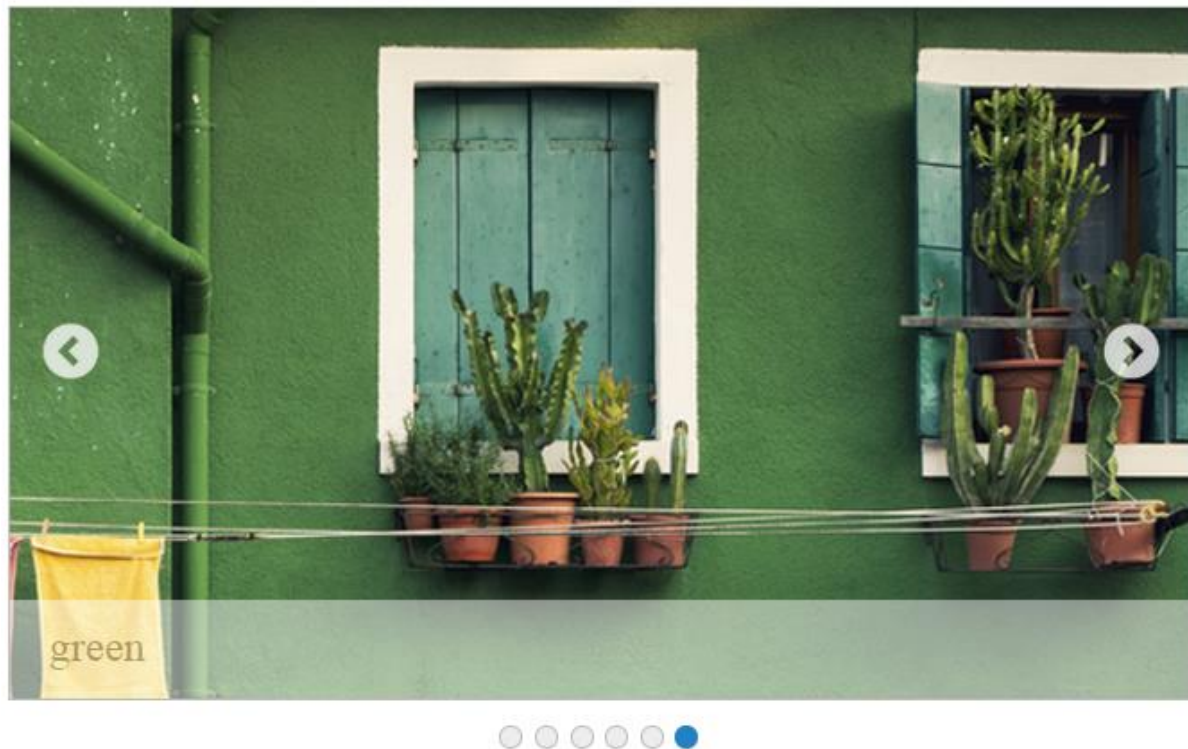
**Rotator** provides the data binding support for the **Rotator** item. So you can bind the data from **JSON Data**. For this behavior, you need to map the corresponding filed with their column names. The data can be bound as a list and it is assigned to **dataSource** property. You can refer the following code example to bind local data.

## HTML

```
<div class="cols-sample-area">
<ul id="sliderContent"></ul>
</div>
```

## JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
$(function () {
var website = [
{ text: "Beautiful Bird", url: "../images/rotator/bird.jpg" },
{ text: "Colorful Night", url: "../images/rotator/night.jpg" },
{ text: "Technology", url: "../images/rotator/tablet.jpg" },
{ text: "Nature", url: "../images/rotator/nature.jpg" },
{ text: "Snow Fall", url: "../images/rotator/snowfall.jpg" },
{ text: "Credit Card", url: "../images/rotator/card.jpg" },
{ text: "Amazing Sculptures", url: "../images/rotator/sculpture.jpg" }
];
var rotatorInstance = new ej.Rotator($("#sliderContent"), {
slideWidth: "600px",
frameSpace: "0px",
displayItemsCount: "1",
slideHeight: "350px",
navigateSteps: "1",
enableResize: true,
pagerPosition: ej.Rotator.PagerPosition.Outside,
dataSource: website,
orientation: ej.Orientation.Horizontal,
showPager: true,
enabled: true,
showCaption: true,
allowKeyboardNavigation: true,
enableRTL: true,
showPlayButton: true,
animationType: "slide"
});
});
}
```



## Behavior settings

### Enabling rotator

The **“enabled”** property enables or disables the **Rotator** control. The default value is **‘true’**. The value set to this property is **Boolean** type. You can refer the following code example to set this property.

### HTML

```
<div class="cols-sample-area">
<ul id="sliderContent" accesskey="e">
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>
```

```

</li>
<li>

</li>
</ul>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
$(function () {
var rotatorInstance = new ej.Rotator($("#sliderContent"), {
enabled: true
});
});
}
```

### Responsive rotator

The “isResponsive” property resizes the Rotator when the browser window is resized. The default value is ‘false’. The value set to this property is Boolean.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
$(function () {
var rotatorInstance = new ej.Rotator($("#sliderContent"), {
isResponsive : true
});
});
}
```

### Auto Play

The **Rotator** Items continuously rotate without user interference by enable the **enableAutoPlay** property. The default value is ‘false’. The value set to this property is **Boolean**.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
$(function () {
var rotatorInstance = new ej.Rotator($("#sliderContent"), {
enableAutoPlay: true
});
});
}
```

### Stop on hover

The **stopOnHover** property pauses the **auto play** while hover on the **Rotator** content. The default value is **'false'**. The value set to this property is **Boolean**.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      enableAutoPlay: true,
      stopOnHover: true
    });
  });
}
```

### Pager settings

#### Pager position

This property specifies the position of the **showPager** in the **Rotator** Item. The default value is **'outside'**. The value set to this property is **string** or **enum**.

- ej.Rotator.PagerPosition.BottomLeft
- ej.Rotator.PagerPosition.BottomRight
- ej.Rotator.PagerPosition.Outside
- ej.Rotator.PagerPosition.TopCenter
- ej.Rotator.PagerPosition.TopLeft
- ej.Rotator.PagerPosition.TopRight

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      pagerPosition: ej.Rotator.PagerPosition.BottomLeft,
      slideWidth: "630px",
      slideHeight: "350px"
    });
  });
}
```



#### Show pager

The “**showPager**” property turns on or off the **pager** support in the **Rotator** control. The **Pager** is used to navigate the **Rotator** Items. The default value is ‘**true**’. The value set to this property is **Boolean**.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      showPager: false
    });
  });
}
```





[Show options](#)

[Show play button](#)

The “**showPlayButton**” property enables play / pause button on **Rotator**. The default value is ‘**false**’. The value set to this property is **Boolean**.

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      slideWidth: "600px",
      showPlayButton: true
    });
  });
}
```



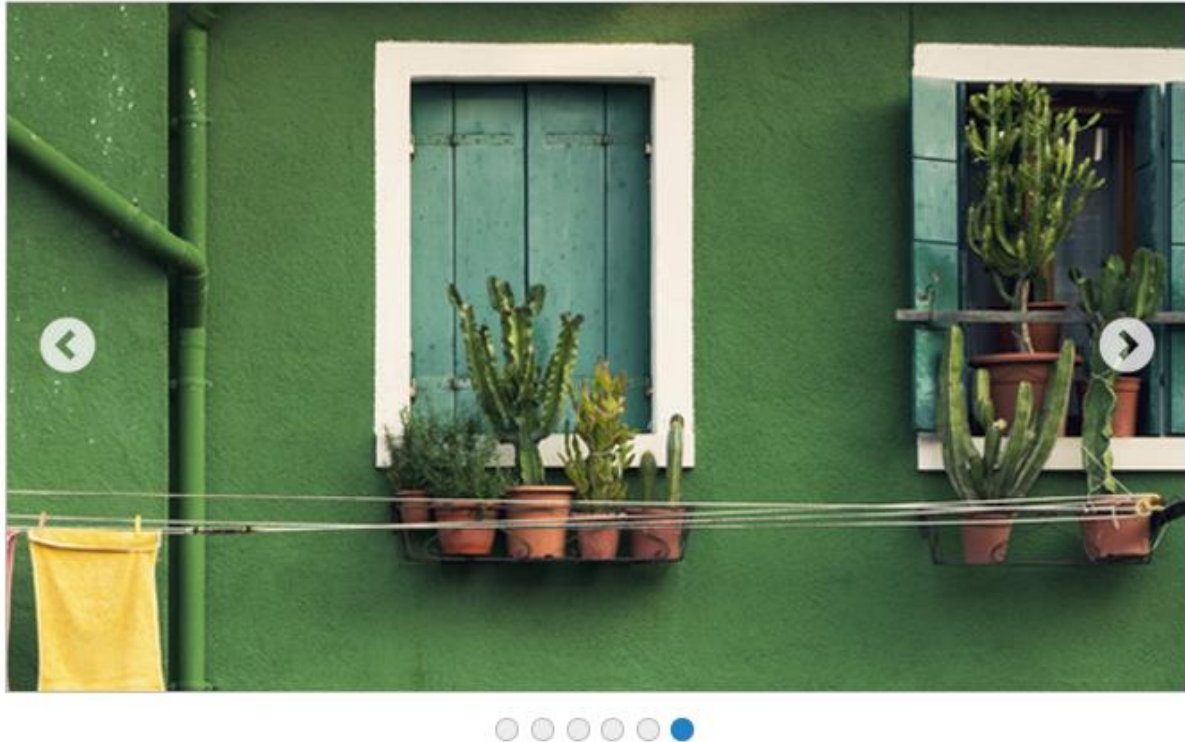
#### Show navigate button

The “**showNavigateButton**” property turns on or off the slide buttons (next and previous) in the **Rotator** Items. Slide buttons are used to navigate the **Rotator** Items. The default value is ‘false’. The value set to this property is **Boolean**.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      slideWidth: "500px",
      showNavigateButton: true
    });
  });
}
```



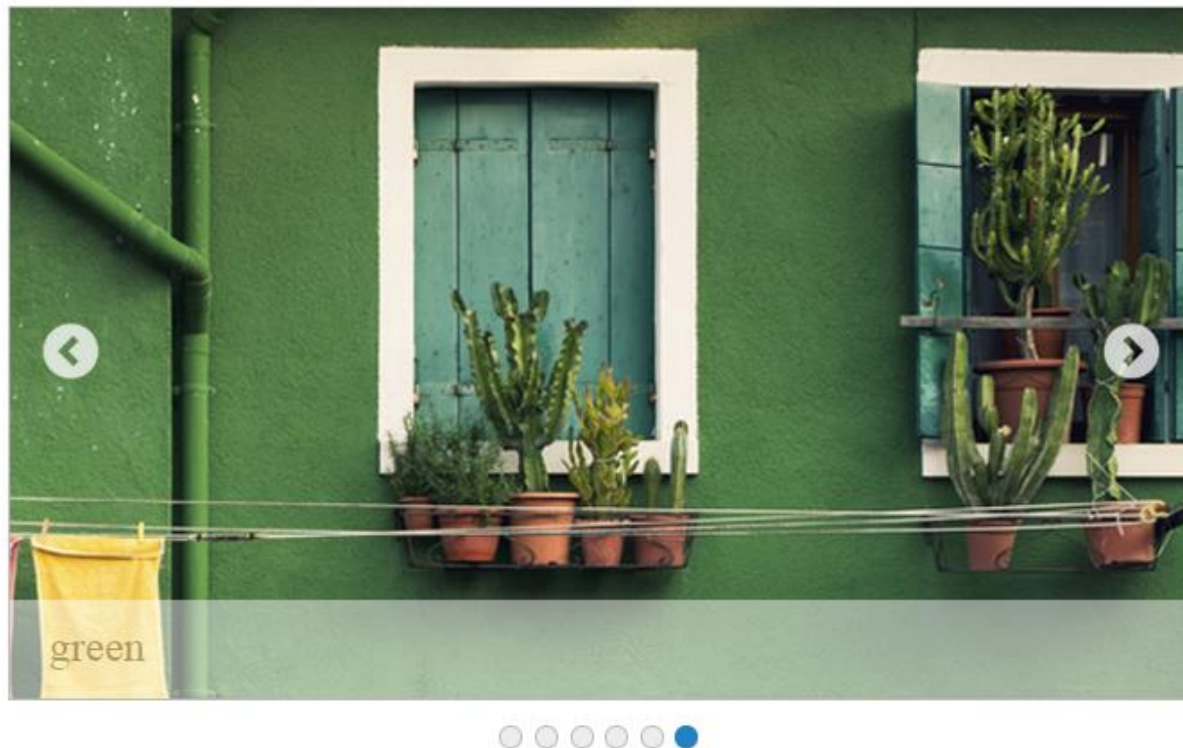


#### Show caption

When the **Rotator** Item is an image, you can specify a caption for the **Rotator** Item. The caption text for each **Rotator** Item is set by using the title attribute of the respective **<image>** tag. The caption cannot be displayed when multiple **Rotator** Items are present. The default value is **'false'**. The value set to this property is **Boolean**.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      slideWidth: "500px",
      showCaption: true
    });
  });
}
```



## Thumbnail

This feature implements **Thumbnail** in **Rotator** control. You can view or access any of the Rotator items instantly. All the images are given as **Thumb Element** to use this feature.

The property **showThumbnail** turns on or off thumbnail support in the **Rotator** control. Thumbnail is used to navigate between slides. Thumbnail supports only single slide transition. You must specify the source for thumbnail elements through the **thumbnailSourceID** property. The default value is **'false'**. The value set to this property is **boolean**.

The property **thumbnailSourceID** specifies the source for thumbnail elements. The default value is **null**. The value set to this property is **object**.

You can refer the following code example of Thumbnail in Rotator.

## HTML

```
<div class="cols-sample-area">
<ul id="sliderContent">
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>
```

```


</li>
<li>

</li>
<li>

</li>
</ul>
<ul id="thumbElement" style="display: none">
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
</ul>
</div>

```

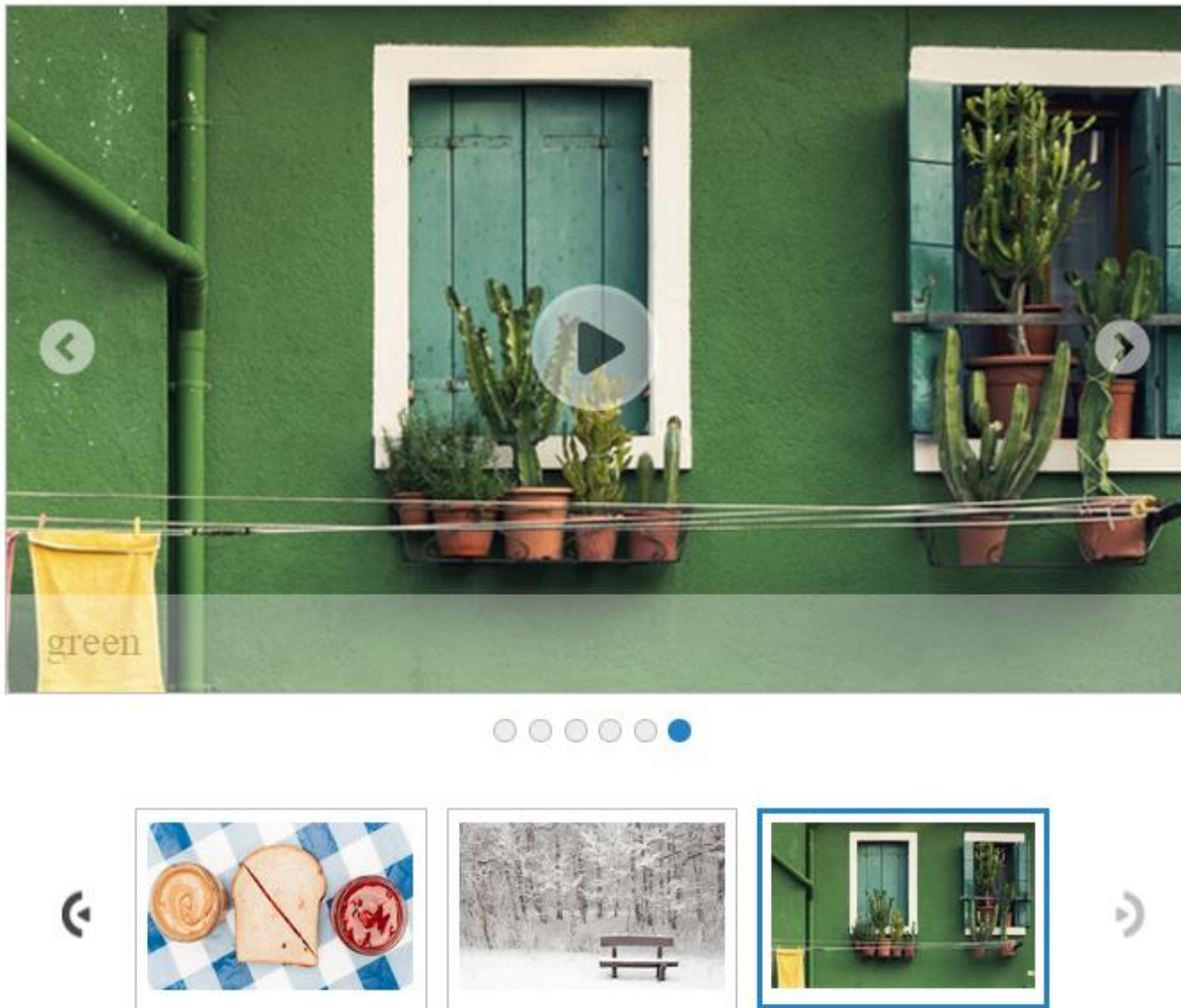
## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
$(function () {
var rotatorInstance = new ej.Rotator($("#sliderContent"), {
slideWidth: "600px",
frameSpace: "0px",
displayItemsCount: "1",
slideHeight: "350px",
navigateSteps: "1",
enableResize: true,
pagerPosition: ej.Rotator.PagerPosition.Outside,
showThumbnail: true,
thumbnailSourceID: "thumbElement",
orientation: ej.Orientation.Horizontal,
showPager: false,
enabled: true,
showCaption: false,
allowKeyboardNavigation: true,

```

```
showPlayButton: true,  
enableAutoPlay: false,  
animationType: "slide"  
});  
});  
}
```



### Orientation

The **Rotator** control supports both vertical and horizontal orientations allowing it to fit into any scenario. The **Rotator** property **orientation** defines the **orientation** where the control is rendered. The value set to this property is **enum or string** type. It accepts the following values.

- ej.Orientation.Horizontal or "Horizontal"
- ej.Orientation.Vertical or "Vertical"

The following steps explain you how to set **orientation** for the **Rotator**.

### Horizontal

This property sets the **Rotator** in **horizontal orientation**. You can refer the following steps to set **horizontal orientation** for **Rotator** control.

In **View**, create ul-li list of **Rotator** items and invoke the **Rotator** helper.

Add the following script in your **HTML** page.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
    $(function () {
        var rotatorInstance = new ej.Rotator($("#sliderContent"), {
            slideWidth: 500,
            orientation: ej.Orientation.Horizontal
        });
    });
}
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
    $(function () {
        var rotatorInstance = new ej.Rotator($("#sliderContent"), {
            slideWidth: 500,
            orientation: "Horizontal"
        });
    });
}
```

### Vertical

This property sets the **Rotator** in **vertical orientation**. You can refer the following steps to set **vertical orientation** for **Rotator** control.

Create ul-li list of **Rotator** items and invoke the **Rotator** helper.

Add the following script in your **HTML** page.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
    $(function () {
        var rotatorInstance = new ej.Rotator($("#sliderContent"), {
            slideWidth: 500,
            orientation: ej.Orientation.Vertical
        });
    });
}
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      slideWidth: 500,
      orientation: "Vertical"
    });
  });
}
```

## Appearance and Styling

### Adjusting rotator item size

#### Slide width

This property sets the **width** of a **Rotator** Item. The value set to this property is **string** or **number**.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />\
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      slideWidth: "500px"
    });
  });
}
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />\
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      slideWidth: 500
    });
  });
}
```

#### Slide height

This property sets the **height** of a **Rotator** Item. The value set to this property is **string** or **number**.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />\
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      slideHeight: "500px"
    });
  });
}
```



## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      slideHeight: 500
    });
  });
}

```

## Image with Contents

This feature allows you to add text along with the **image** in **Rotator** control. This is achieved by splitting the content into two panels. In the following code example, image is given in the left panel and text is given in the right panel.

## HTML

```

<html>
<body>
<div class="cols-sample-area">
<ul id="sliderContent">
<li>
<div class="leftPanel">

</div>
<div class="rightPanel blog">
<div class="contentPanel">Tablet </div>
<ul>
<li>A tablet computer, or simply tablet, is a mobile computer with
display, circuitry and battery in a single unit.</li>
<li>
Tablets are equipped with sensors, including cameras, microphone,
accelerometer and touchscreen,
</ul>
</div>
</li>
<li>
<div class="leftPanel">

</div>
<div class="rightPanel">
<div class="contentPanel">Nature </div>
<ul>
<li>The health of the natural environment is critical to the long-term
future of the planet</li>
<li>Nature, in the broadest sense, is equivalent to the natural,
physical, or material world or universe.</li>
</ul>
</div>
</li>
<li>
<div class="leftPanel">

```

```


</div>
<div class="rightPanel credit">
<div class="contentPanel">Credit card </div>
<ul>
<li>A credit card is a payment card issued to users as a system of
payment</li>
<li>It allows the card holder to pay for goods and services based on the
holder's promise to pay for them</li>
</ul>
</div>
</li>
<li>
<div class="leftPanel">

</div>
<div class="rightPanel">
<div class="contentPanel">Rose </div>
<ul>
<li>A rose is a woody perennial of the genus Rosa, within the family
Rosaceae</li>
<li>Flowers vary in size and shape and are usually large and showy,
There are over 100 species
</li>
</ul>
</div>
</li>
<li>
<div class="leftPanel">

</div>
<div class="rightPanel rightSide">
<div class="contentPanel">Snowfall </div>
<ul>
<li>Mt. Baker ski area in Washington State has the world record for
snowfall at 1,140 inches of snow in the 1998/1999 winter season</li>
<li>Mt. Baker ski area is located near but not on the real 10,781 Mount
Baker</li>
</ul>
</div>
</li>
</ul>
</div>
</body>
</html>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
$(function () {
var rotatorInstance = new ej.Rotator($("#sliderContent"), {
slideWidth: "600px",
displayItemCount: "1",
slideHeight: "300px",

```

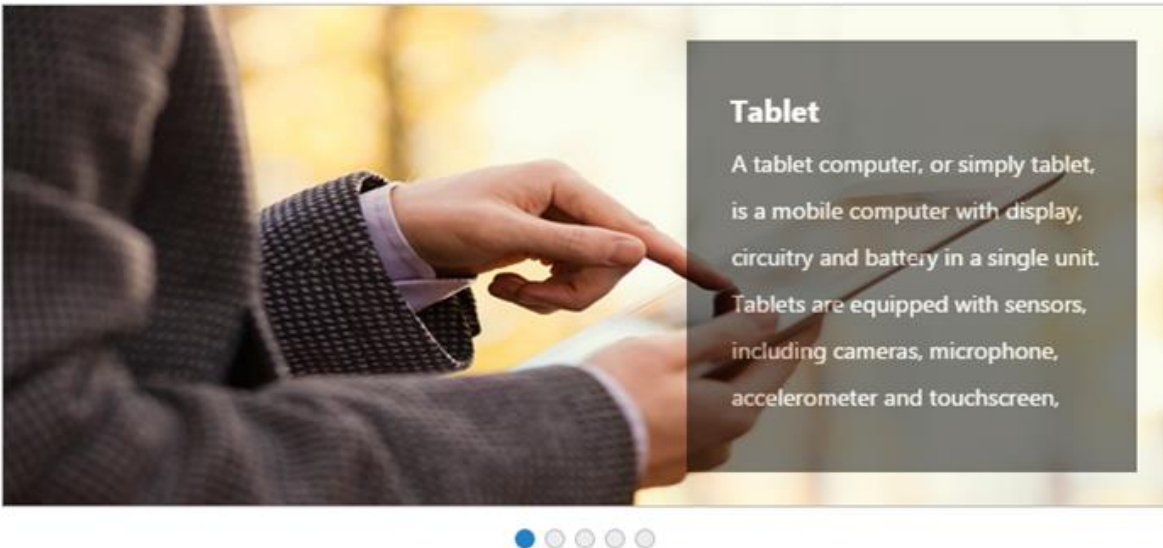


```
navigateSteps: "1",
enableResize: true,
pagerPosition: ej.Rotator.PagerPosition.Outside,
orientation: ej.Orientation.Horizontal,
showPager: true,
enabled: true,
showCaption: false,
allowKeyboardNavigation: true,
showPlayButton: true,
animationType: "slide",
enableRTL: true
});
});
}
```

## CSS

```
<style type="text/css" class="cssStyles">
#sliderContent > li {
background-color: #9ee8d8;
}
#sliderContent > li .leftPanel {
float: left;
width: 700px;
height: 300px;
padding-right: 0px;
}
#sliderContent > li .leftPanel img {
width: 700px;
height: 300px;
}
#sliderContent .rightPanel {
background-color: #FFFFFF;
height: 259px;
margin-left: 410px;
margin-top: 21px;
opacity: 0.5;
padding-left: 10px;
position: absolute;
width: 260px;
}
#sliderContent .rightPanel.credit {
opacity: 0.6;
}
#sliderContent .rightPanel.blog {
background-color: black;
}
#sliderContent .rightPanel.blog li {
color: white;
list-style: none;
line-height: 2;
}
#sliderContent .rightPanel.blog .contentPanel {
padding-top: 30px;
color: white;
}
```

```
#sliderContent .rightPanel .contentPanel {
color: #000000;
font-size: large;
font-weight: bold;
left: 16px;
padding-top: 30px;
position: relative;
}
#sliderContent .rightPanel li {
color: black;
list-style: none;
line-height: 2;
}
#sliderContent .rightPanel.rightSide {
margin-left: 20px;
background-color: black;
}
#sliderContent .rightPanel.rightSide li {
color: white;
list-style: none;
line-height: 2;
}
#sliderContent .rightPanel.rightSide .contentPanel {
padding-top: 30px;
color: white;
}
.rightPanel > ul {
padding: 6px 17px 17px;
}
</style>
```



Display items

*Display Items count*

This property specifies the number of **Rotator** Items to be displayed. The default value is **'1'**. The value set to this property is **string** or **number**.

**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
$(function () {
var rotatorInstance = new ej.Rotator($("#sliderContent"), {
slideWidth: "200px",
displayItemsCount: 3,
slideHeight: "165px"
});
});
}

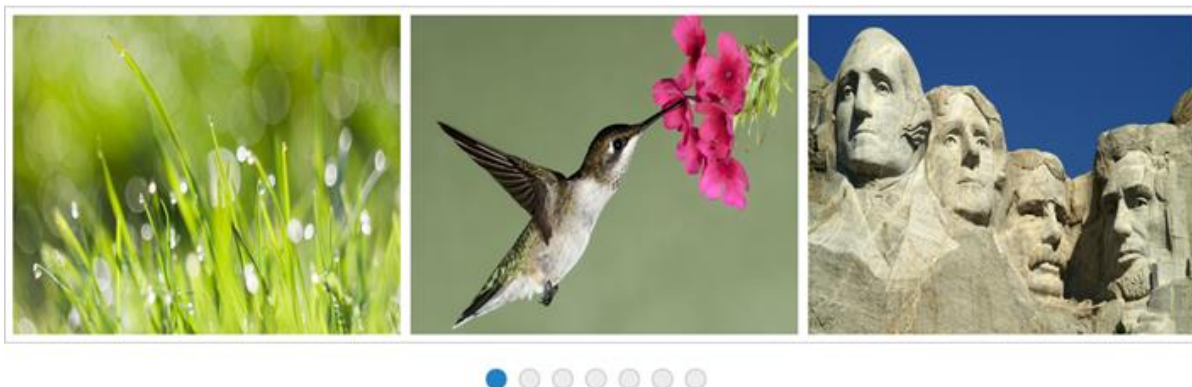
```

**CSS**

```

<style type="text/css" class="cssStyles">
#sliderContent > li .image {
width:200px;
height:165px;
}
</style>

```

*Navigate steps*

This property specifies the number of **Rotator** Items to **navigate** on a single click (next/previous/play buttons). The **navigateSteps** property value must be less than or equal to the **displayItemsCount** property value. The default value is '1'. The value set to this property is **string** or **number**.

**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
$(function () {
var rotatorInstance = new ej.Rotator($("#sliderContent"), {
slideWidth: 200,
slideHeight: 165,
displayItemsCount: 2,
navigateSteps: 2
});
});
}

```

## CSS

```
<style type="text/css" class="cssStyles">
#sliderContent > li .image {
width:200px;
height:165px;
}
</style>
```

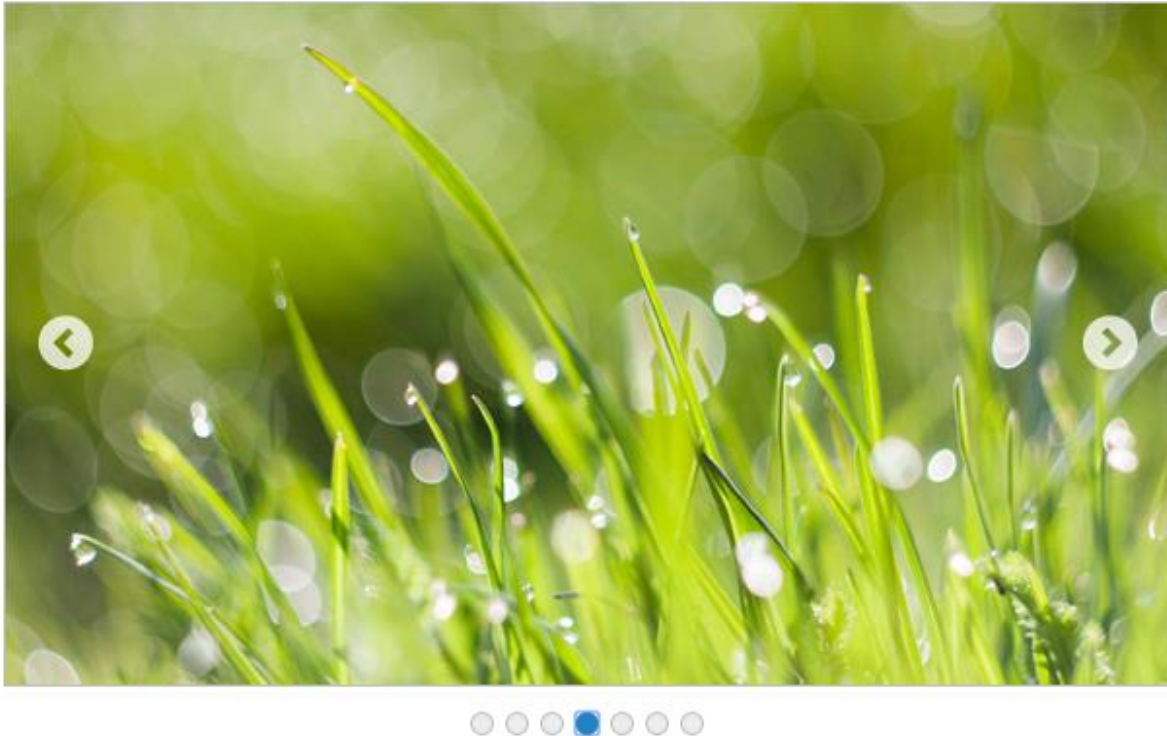


### [Start index](#)

This property sets the **index** of the slide that is displayed first. The default value is '1'. The value set to this property is **string** or **number**.

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
$(function () {
var rotatorInstance = new ej.Rotator($("#sliderContent"), {
slideWidth: 500,
startIndex: 4
});
});
}
```



#### Frame space

This property sets the space between the **Rotator** Items. The value set to this property is **string** or **number**.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
$(function () {
var rotatorInstance = new ej.Rotator($("#sliderContent"), {
slideWidth: "500px",
slideHeight: "300px",
displayItemCount: 2,
frameSpace: "30px"
});
});
}
```

#### Animation

**animationType** property specifies the **Animation** type for the **Rotator** Item. **animationType** options include slide, fastSlide, slowSlide, and other custom easing animationTypes. The default value is 'slide'. The value set to this property is **string**.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
$(function () {
```

```
var rotatorInstance = new ej.Rotator($("#sliderContent"), {
  slideWidth: "500px",
  animationType: "slowSlide"
});
});
}
```

#### Animation speed

This property sets the **speed** of slide transition. The default value is '600'. The value set to this property is **string** or **number**.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      slideWidth: "500px",
      animationSpeed: 3000
    });
  });
}
```

#### Delay

This property sets the **delay** between the **Rotator** Items to move after the slide transition. The default value is 500. The value set to this property is **string** or **number**.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      slideWidth: "500px",
      delay: 5000
    });
  });
}
```

#### Theme

**Rotator** control's style and appearance are controlled based on **CSS classes**. In order to apply styles to the **Rotator** control, you can refer 2 files namely, **ej.widgets.core.min.css** and **ej.theme.min.css**. When the file **ej.widgets.all.min.css** is referred, then it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.widgets.all.min.css** is the combination of these both.

By default, there are 17 theme support available for RadialMenu component, namely:

- flat-azure
- flat-azure-dark
- fat-lime

- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap
- high-contrast-01
- high-contrast-02
- material
- office-365

#### *cssClass*

This property is used to set **root class** for **Rotator** control theme. The value set to this property is **string** type.

#### **JAVASCRIPT**

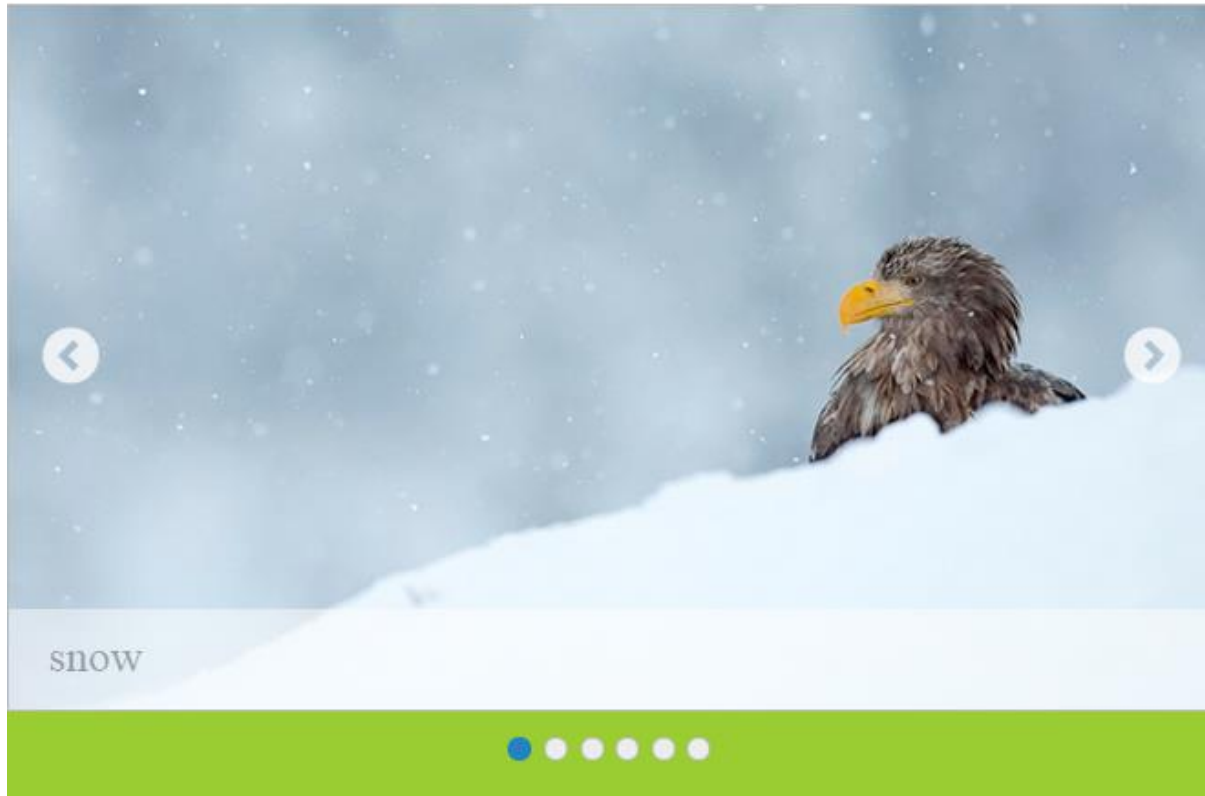
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      slideWidth: 500,
      slideHeight: 300,
      cssClass: "flat-lime"
    });
  });
}
```

Add the following code in your **Css**.

#### **CSS**

```
<style>
.flat-lime {
  background-color: yellowgreen;
}
</style>
```





### RTL Support

This feature supports to change the left-to-right alignment of the **Rotator** to right-to-left (RTL). (I.e.) Sets the **Rotator** to do its actions from right to left. The property **enableRTL** sets the **Rotator** from right to left. The value set to this property is **boolean** type.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
  $(function () {
    var rotatorInstance = new ej.Rotator($("#sliderContent"), {
      slideWidth: 500,
      enableRTL: true
    });
  });
}
```

### Keyboard interaction

The **Rotator** property **allowKeyboardNavigation** turns on keyboard interaction with the Rotator items. You must set this property to **'true'** to access the keyboard shortcuts. The default value is **'true'**. The value set to this property is **Boolean**.

The entire Rotator commands are accessed through the keyboard by specifying the **Keyboard Shortcut** in the following table.

Keyboard shortcuts



| KeyboardShortcut | Function                        |
|------------------|---------------------------------|
| Alt + j          | Focuses the control             |
| UP               | Navigates up and left.          |
| Down             | Navigates down and right.       |
| Left             | Navigates up and left.          |
| Right            | Navigates down and right.       |
| Home             | Navigates to the starting item. |
| End              | Navigates to the ending item.   |
| Enter            | Selects the focused item        |

You can refer the following code example for **keyboard** navigation.

### HTML

```
<div class="cols-sample-area">
<ul id="sliderContent" accesskey="e">
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
</ul>
<ul id="slide" style="display: none">
<li>

</li>
```

```

<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
<li>

</li>
</ul>
</div>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module RotatorComponent {
$(function () {
var rotatorInstance = new ej.Rotator($("#sliderContent"), {
slideWidth: "550px",
frameSpace: "0px",
displayItemsCount: "1",
slideHeight: "350px",
navigateSteps: "1",
enableResize: true,
pagerPosition: ej.Rotator.PagerPosition.Outside,
showThumbnail: true,
thumbnailSourceID: "slide",
orientation: ej.Orientation.Horizontal,
enableRTL: true,
showPager: false,
enabled: true,
showCaption: false,
allowKeyboardNavigation: true,
showPlayButton: true,
animationType: "slide",
});
});
}

```

Add the following code in your **JavaScript** to focus the control.

## JAVASCRIPT

```

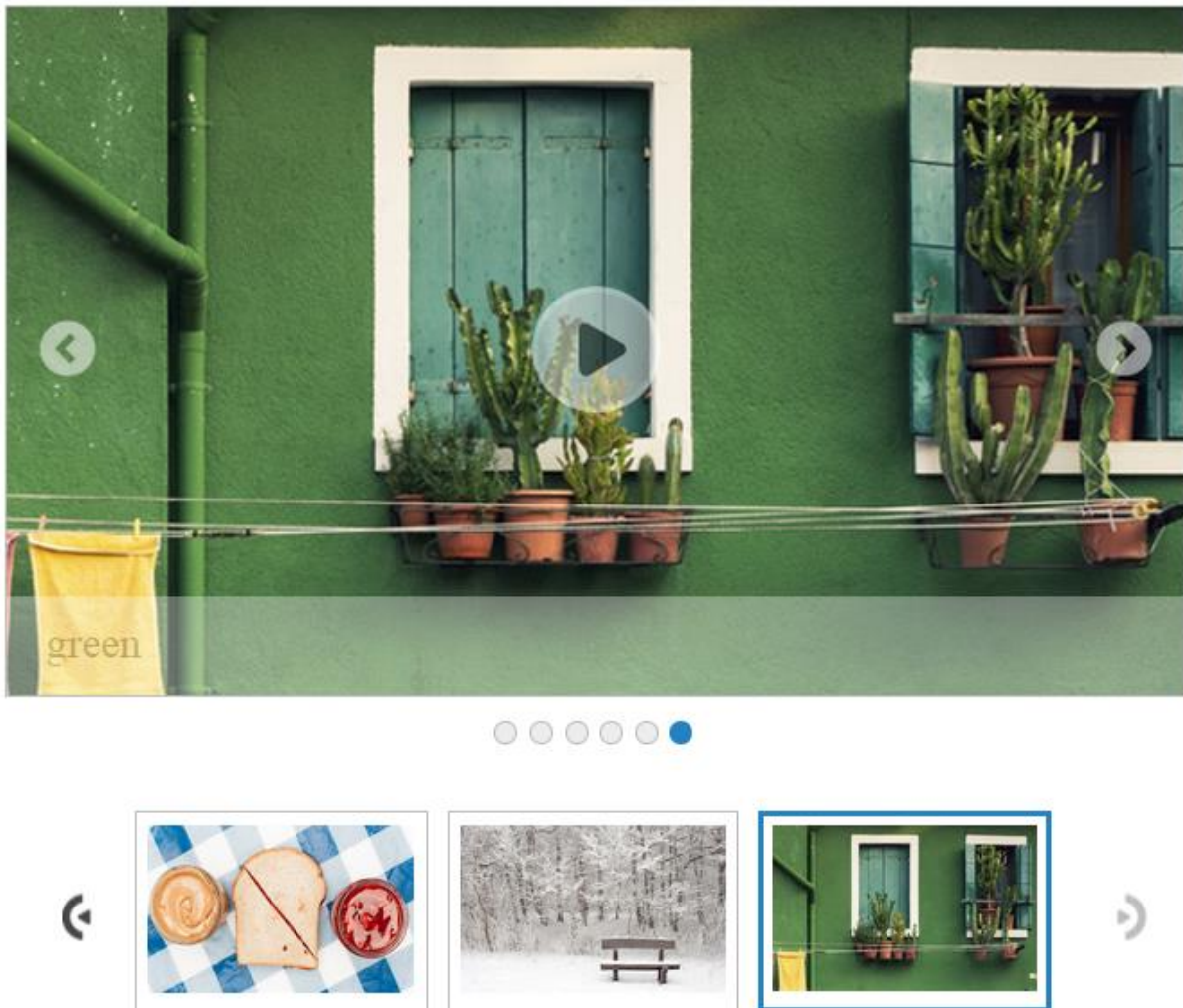
//Control focus key
$(document).on("keydown", function (e) {
if (e.altKey && e.keyCode === 74) { // j- key code.

```

```
$("#sliderContent")[0].focus();
}
});
```

## CSS

```
<style type="text/css" class="cssStyles">
.e-rotator-wrap .e-thumb .e-thumb-items li img {
width: 130px;
height: 82px;
}
</style>
```



## Schedule

### Overview

**Scheduler** is an event calendar that manages the list of various activities (events/appointments) in different available views (day, week, workweek, month and agenda) for various resources. It is mainly

for tracking the user appointments and allows them to create a new or edit/delete the older ones. Almost all the features in JS Scheduler are applicable to Scheduler in TypeScript too.

### Key Features

Some of the key features of Scheduler are as follows,

- **DataSource** - Supports various client-side and remote data sources such as JSON, RESTful services, OData services, WCF services and much more.
- **Views** - 6 types of views are available namely Day, Week, WorkWeek, Month, Agenda and Custom View (user-specific date rendering).
- **Adaptive** - Scheduler UI layout adapts automatically according to the desktop/mobile mode (responsive support).
- **Resize & Drag** - Appointments can be resized and dragged anywhere within the Scheduler. External drag and drop of appointments are also applicable now.
- **Multiple Resources & Grouping** - Allows the Scheduler to categorize and display resources in a hierarchical structure either in a horizontal or vertical manner.
- **Orientation** - Two types of control orientation is supported namely – Vertical and Horizontal (Timeline View).
- **Categories** - 6 default types of Appointment categories along with user customization options are available to differentiate the appointment status.
- **Template** - Template customization provided for appointments, resource header, cells, date header, priority, tooltip, time scale and agenda view.
- **TimeZone & DST** - Supports observation of Daylight Saving Time in Scheduler for whichever time zone it is applicable.
- **Export & Print** - Supports exporting of single/all appointment(s) to an ICS file and also Prints all/specific appointment(s).
- **PDF Export** - Supports exporting of entire Scheduler into PDF format.
- **Appointment window Customization** – Entire appointment window can be customized with the user required fields.
- **Recurrence Editor** - Complete recurrence related options of Scheduler are collectively defined as a separate plug-in, which can be utilized directly within the customized appointment window.

### External and Internal Dependencies

The following list of external dependencies are mandatory in order to render any of the Syncfusion controls -

- [jQuery](#) - 1.7.1 and later versions
- [jsRender](#) - to render the templates

The following list of external dependencies are also needed to render the Schedule in Typescript.

- `jquery.d.ts` - Definition file that tells the Typescript, how the JavaScript library works. Download from [here](#).
- `ej.web.all.d.ts` - Type definition file that helps to render the Syncfusion widgets in Typescript.

---

**Information:** `ej.web.all.d.ts` file is available under the location, *(Installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript*.

---

The other required internal dependencies of Scheduler are tabulated below,

| File  | Description/Usage   |
|---|---|
| ej.core.min.js  | Must be referred always first before using all the JS controls.   |
| ej.data.min.js  | Used to handle data operation and should be used while binding data to JS controls.   |
| ej.globalize.min.js   | Must be referred to localize any of the JS control's text and content.  |
| ej.schedule.min.js  | Schedule core script file which includes schedule related scripts files such as ej.schedule.render.js, ej.schedule.resources.js and ej.schedule.horizontal.js |
| ej.scroller.js<br>ej.touch.js<br>ej.draggable.js<br>ej.navigationdrawerbase.js<br>ej.listviewbase.js<br>ej.listview.js<br>ej.recurrenceeditor.js<br>ej.dropdownlist.js<br>ej.radiobutton.js<br>ej.dialog.js<br>ej.button.js<br>ej.autocomplete.js<br>ej.datepicker.js<br>ej.timepicker.js<br>ej.checkbox.js<br>ej.editor.js<br>ej.menu.js | These files are referred for proper working of the sub-controls used within Scheduler.  |

|                        |  |
|------------------------|--|
| ej.navigationdrawer.js |  |
| ej.tooltip.js          |  |

**Note:** Scheduler uses one or more sub-controls, therefore refer the `ej.web.all.min.js` (which encapsulates all the `ej` controls and frameworks in a single file) in the application instead of referring all the above specified internal dependencies.

To get the real appearance of the Scheduler, the dependent CSS file `ej.web.all.min.css` (which includes styles of all the widgets) should also needs to be referred.

**Note:** Uncompressed version of library files are also available which is used for development or debugging purpose and can be generated from the custom script [here](#).

## Getting Started

To get start with how to create a general Typescript application and use Syncfusion widgets within it, refer [here](#). To know more about the Scheduler individual script references, refer [here](#).

### Create an HTML File

Create a new HTML file and include the below initial code.

#### HTML

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title> </title>
</head>
<body>
</body>
</html>
```

## Script/CSS References

Refer the CSS file from the specific theme folder to your HTML file within the `head` section. Refer the built-in available themes from [here](#).

#### HTML

```
<head>
<meta charset="utf-8" />
<title>Getting Started - Schedule</title>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />
</head>
```

Add links to the [CDN](#) Script files of the other required external dependencies.

#### HTML

```
<head>
<meta charset="utf-8" />
<title>Getting Started - Schedule</title>
```

```
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"></scri
pt>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"></script>
<script src="app.js"></script>
</head>
```

**Note:** Uncompressed version of library files are also available which is used for development or debugging purpose and can be generated from the custom script [here](#).

**Note:** The content of `app.js` file reference in the above script section is automatically generated from the `app.ts` file, when the typescript application is compiled.

### Define Container to render Scheduler

Create a `div` element within the body section of the HTML document, where the Scheduler needs to be rendered.

### HTML

```
<body>
<div id="Schedule1"></div>
</body>
```

### Control Initialization

Create a typescript module in `app.ts` file with the control initialization code for rendering the scheduler control in its container namely `Schedule1` created through the HTML file. Before the module creation, make sure to refer the `ej.web.all.d.ts` and `jquery.d.ts` type-definition files in your project in order to support the typescript rendering.

### TS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var dManager = [{
Id: 1,
Subject: "Bering Sea Gold",
StartTime: new Date(2014, 4, 5, 5, 30),
EndTime: new Date(2014, 4, 5, 7, 30),
Description: "",
AllDay: false,
Recurrence: false
}];
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
height: "525px",
currentDate: new Date(2014, 4, 5),
appointmentSettings: {
```

```

dataSource: dManager,
id: "Id",
subject: "Subject",
startTime: "StartTime",
endTime: "EndTime",
description: "Description",
allDay: "AllDay",
recurrence: "Recurrence",
recurrenceRule: "RecurrenceRule"
}
});
});
}

```

Now build your application, so that the `app.js` file is automatically generated from the above typescript file and got added to your project. Now, whatever code changes that you make in `app.ts` file will be reflected in `app.js` file automatically on every successful compilation.

### Setting TimeZone

Initially, the system timezone is preferred as the Schedule's default timezone. When some specific timezone is explicitly defined to the Schedule, it will be set to it.

Likewise, we can also set different timezones for the appointments. Usually, the system timezone is assigned as the appointment's default timezone and it will be positioned on the Scheduler based on the start/end time range and timezone assigned to it.

To set Scheduler `timeZone` and its appointment `timeZone` values, refer the below code example -

### TS

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var dManager = [{
Id: 1,
Subject: "Bering Sea Gold",
StartTime: new Date(2014, 4, 5, 5, 30),
StartTimeZone: "UTC +02:00",
EndTime: new Date(2014, 4, 5, 7, 30),
EndTimeZone: "UTC +02:00",
Description: "",
AllDay: false,
Recurrence: false
}];
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
height: "525px",
currentDate: new Date(2014, 4, 5),
timeZone: "UTC +05:30",
appointmentSettings: {
dataSource: dManager,
id: "Id",
subject: "Subject",
startTime: "StartTime",
startTimeZone: "StartTimeZone",

```



```

endTime: "EndTime",
endTimeZone: "EndTimeZone",
description: "Description",
allDay: "AllDay",
recurrence: "Recurrence",
recurrenceRule: "RecurrenceRule"
}
});
});
}

```

## Data Binding

### Appointment Fields

The below listed names are the appointment fields which holds the appropriate column names from the dataSource.

| Field name    | Description   |
|---------------|---|
| id            | Binds the id field name for indexing and performing CRUD operation on the appointments. It's optional.  |
| startTime     | Binds the appointment start time field name which is mandatory and also its related validation rules.   |
| startTimeZone | Binds the name of the start timezone field in the dataSource and also its related validation rules. If the startTimeZone field is not mentioned, then the appointment makes use of the Scheduler timeZone or System timeZone.   |
| endTime       | Binds the appointment end time field name which is mandatory and also its related validation rules.   |
| endTimeZone   | Binds the name of the end timezone field in the dataSource and also its related validation rules. If the endTimeZone field is not mentioned, then the appointment makes use of the Scheduler timeZone or System timeZone.       |
| subject       | Binds the appointment subject field name which holds the summary of the appointment and also its related validation rules.  |
| location      | Binds the name of the location field and also its related validation rules. It indicates the appointment location/occurrence place. This field needs to be bind to the Scheduler, when an API showLocationField is set to true. |
| description   | Binds the appointment description field name and also its related validation rules.   |
| allDay        | Binds the name of the <code>allDay</code> field. It accepts the boolean value and indicates whether the appointment is an all-day appointment or not.   |

|                  |   |
|------------------|---|
| categorize       | Binds the name of the categorize field and also its related validation rules. It indicates the category or status value (red categorize, green, yellow and so on).  |
| priority         | Binds the name of the priority field, its related validation rules and also indicates the priority (high, low, medium and none) of the appointments. This field should be bind to the Scheduler, when prioritySettings.enable is set to true. |
| resourceFields   | Binds one or more fields in the resource collection. It maps the resource field names with the appointments, denoting to which resource the appointments actually belongs.  |
| recurrence       | Binds the name of the recurrence field. It accepts the boolean value and indicates whether the appointment is a recurrence appointment or not.  |
| recurrenceRule   | Binds the name of the recurrenceRule field. It holds the recurrence pattern associated with the appointments.   |
| recurrenceId     | Binds the recurrence Id field which acts as a parent id for Scheduler recurrence appointments.  |
| recurrenceExDate | Binds the recurrence Exception field which accepts the recurrence Exception date values.  |

The below example depicts the appointment fields accepting the string type mapper fields,

#### HTML

```
<!-- HTML element will initialize as a ejSchedule -->
<div id="schedule"></div>
```

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
currentDate: new Date(2015, 11, 7),
showLocationField: true,
categorizeSettings: {
enable: true
},
prioritySettings: {
enable: true
},
group: {
resources: ["Owners"]
},
resources: [{
field: "ownerId",
```

```

title: "Owner",
name: "Owners",
resourceSettings: {
  dataSource: [{
    text: "Nancy",
    id: 1,
    color: "#f8a398"
  }, {
    text: "Steven",
    id: 3,
    color: "#56ca85"
  }, {
    text: "Michael",
    id: 5,
    color: "#51a0ed"
  }],
  text: "text",
  id: "id",
  color: "color"
},
appointmentSettings: {
  resourceFields: "ownerId",
  dataSource: [{
    Id: 1,
    Subject: "Music Class",
    StartTime: new Date("2015/11/7 06:00 AM"),
    StartTimeZone: "UTC +05:30",
    EndTime: new Date("2015/11/7 07:00 AM"),
    EndTimeZone: "UTC +05:30",
    Description: "Never Give up on Obstacles",
    location: "US",
    AllDay: false,
    Recurrence: true,
    RecurrenceRule: "FREQ=WEEKLY;BYDAY=MO,TU;INTERVAL=1;COUNT=15",
    Categorize: "1",
    Priority: "medium",
    ownerId: 3,
    RecurrenceId: 1,
    RecurrenceExDate: null
  }]
}
});
});
}

```

### Appointment Field Validation

It is possible to validate the required fields of the appointment window from client-side before submitting it, by adding appropriate validation rules to each fields. The appointment fields have been extended to accept both String and object type values. Therefore, in order to perform validations, it is necessary to specify object values for the appointment fields.

Refer the appointment fields specified with validation rules from the following code example.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

## JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
$.validator.addMethod("customRule", function (value, element, options) {
var expression = /^[a-zA-Z0-9- ]*$/; //new RegExp(options);
return expression.test(value);
}, "Special character(s) not allowed in Location field");
var dManager =
ej.DataManager(window.Default).executeLocal(ej.Query().take(10));
var sample = new ej.Schedule($("#schedule"), {
width: "100%",
height: "525px",
currentDate: new Date(2014, 4, 5),
showLocationField: true,
categorizeSettings: {
enable: true,
allowMultiple: false,
dataSource: [
{ text: "Blue Category", id: 1, color: "#43b496", fontColor: "#ffffff" },
{ text: "Green Category", id: 2, color: "#7f993e", fontColor: "#ffffff"
},
{ text: "Orange Category", id: 3, color: "#cc8638", fontColor: "#ffffff"
},
{ text: "Purple Category", id: 4, color: "#ab54a0", fontColor: "#ffffff"
},
{ text: "Red Category", id: 5, color: "#dd654e", fontColor: "#ffffff" },
{ text: "Yellow Category", id: 6, color: "#d0af2b", fontColor: "#ffffff"
}
],
text: "text", id: "id", color: "color", fontColor: "fontColor"
},
appointmentSettings: {
id: "Id",
subject: { field: "Subject", validationRules: { required: true } },
location: { field: "Location", validationRules: { required: true,
customRule: "/^[a-zA-Z0-9- ]*$/ " } },
startTime: { field: "StartTime", validationRules: { required: true } },
endTime: { field: "EndTime", validationRules: { required: true } },
description: { field: "Description", validationRules: { required: true,
minlength: 5, maxlength: 500 } },
allDay: "AllDay",
recurrence: "Recurrence",
recurrenceRule: "RecurrenceRule",
categorize: { field: "Categorize", validationRules: { required: true,
messages: { required: "Categories are required." } } }
}
});
});
}
```

### Binding to JSON Data Array

To bind the Scheduler events data as array of JSON objects, refer the below code example.

#### Example - Array of JSON Data Binding

##### HTML

```
<!-- HTML element will initialize as a ejSchedule -->
<div id="schedule"></div>
```

##### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
currentDate: new Date(2015, 11, 7),
appointmentSettings: {
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 06:00 AM"),
EndTime: new Date("2015/11/7 07:00 AM")
}, {
Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 9:00 AM"),
EndTime: new Date("2015/11/7 02:30 PM")
}]
}
});
});
}
```

### Binding Remote Data Service

The appointment data can be bound to the Scheduler through the [Odata](#) remote services, where the service URL is mapped with [Data manager](#) and then configured to the Schedule dataSource API.

##### HTML

```
<!-- HTML element will initialize as a ejSchedule -->
<div id="schedule"></div>
```

##### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var dataManager = ej.DataManager({
url: "http://mvc.syncfusion.com/OdataServices/Northwnd.svc/" // referring
data from remote service (url binding)
});
});
}
```

```

var queryEvent = ej.Query().from("Events").take(10); // query to fetch
the records from the specified table "Events"
var sample = new ej.Schedule($("#schedule"), {
  currentDate: new Date(2014, 4, 5),
  appointmentSettings: {
    // Configure the dataSource with dataManager object
    dataSource: dataManager,
    query: queryEvent
  }
});
});
}

```

## OData V4

The OData v4 is an improved version of OData protocols and the DataManager can also retrieve and consume appointment data from [OData v4](#) services.

### HTML

```

<!-- HTML element will initialize as a ejSchedule -->
<div id="schedule"></div>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    // get the appointments data from OData v4 service
    var dataManager = ej.DataManager({
      url: "http://services.odata.org/V4/Northwind/Northwind.svc/Orders/",
      //OData v4 service
      adaptor: new ej.ODataV4Adaptor()
    });
    var sample = new ej.Schedule($("#schedule"), {
      currentDate: new Date(1997, 2, 23),
      appointmentSettings: {
        // Configure the dataSource with dataManager object
        dataSource: dataManager,
        subject: "ShipName",
        startTime: "OrderDate",
        endTime: "RequiredDate",
        description: "ShipAddress"
      }
    });
  });
}

```

## WebAPI Binding

The Schedule appointment data can be bound through the Web API service and it is a programmatic interface to define the request and response messages system that is mostly exposed in **JSON** or **XML**.

### HTML

```

<!-- HTML element will initialize as a ejSchedule -->

```

```
<div id="schedule"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var dataManager = ej.DataManager({
      // get the required appointments from Web API service
      url: "http://mvc.syncfusion.com/OdataServices/api/ScheduleData/",
      // enable cross domain
      crossDomain: true
    });
    var sample = new ej.Schedule($("#schedule"), {
      currentDate: new Date(2014, 4, 5),
      appointmentSettings: {
        // Configure the dataSource with dataManager object
        dataSource: dataManager
      }
    });
  });
}
```

The server-side code to retrieve the appointments are as follows.

### C#

```
// To retrieve the appointments from database and bind it to Scheduler
public IEnumerable<Event> GetData(String CurrentDate, String CurrentView,
String CurrentAction)
{
    return new NORTHWNDEntities().Events.ToList();
}
```

### Data binding using OLEDB

The appointment data can also be bound to the Scheduler using OLEDB database as depicted below.

### HTML

```
<!-- HTML element will initialize as a ejSchedule -->
<div id="schedule"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    // get the appointment data from the specified controller action
    var dataManager = ej.DataManager({
      url: "Home/GetData", // This will trigger to bind the appointment data
                           // initially to the Schedule control
      crudUrl: "Home/Batch", // This will trigger while performing CRUD
                             // operation on the Scheduler appointments
    });
  });
}
```

```

adaptor: new ej.UrlAdaptor()
});
var sample = new ej.Schedule($("#schedule"), {
currentDate: new Date(2014, 4, 5),
appointmentSettings: {
// Configure the dataSource with dataManager object
dataSource: dataManager,
id: "Id",
subject: "Subject",
startTime: "StartTime",
endTime: "EndTime",
startTimeZone: "StartTimeZone",
endTimeZone: "EndTimeZone",
description: "Description",
allDay: "AllDay",
recurrence: "Recurrence",
recurrenceRule: "RecurrenceRule"
}
});
});
}

```

The server-side controller code to retrieve and bind the appointment data to Scheduler are as follows. Also, define a class with all the required appointment fields as depicted in the below code example.

#### C#

```

// Define a class with all appointment fields
public class ScheduleData
{
public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public Boolean AllDay { get; set; }
public Boolean Recurrence { get; set; }
public string RecurrenceRule { get; set; }
public string StartTimeZone { get; set; }
public string EndTimeZone { get; set; }
public string Description { get; set; }
}
// To retrieve the appointments from database and bind it to Scheduler
public JsonResult GetData()
{
// Mention your own dataSource to be used here.
string strAccessConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=|DataDirectory|/ScheduleDb.MDB";
DataSet myDataSet = new DataSet();
OleDbConnection myAccessConn = null;
myAccessConn = new OleDbConnection(strAccessConn);
OleDbCommand myAccessCommand = new OleDbCommand("SELECT * FROM
DefaultSchedule", myAccessConn);
OleDbDataAdapter myDataAdapter = new OleDbDataAdapter(myAccessCommand);
myAccessConn.Open();
myDataAdapter.Fill(myDataSet, "DefaultSchedule");
List<ScheduleData> datasource = new List<ScheduleData>();

```



```

datasource = myDataSet.Tables[0].AsEnumerable().Select(dataRow => new
ScheduleData { Id = dataRow.Field<int>("Id"), Subject =
dataRow.Field<string>("Subject"), StartTime =
dataRow.Field<DateTime>("StartTime"), EndTime =
dataRow.Field<DateTime>("EndTime"), AllDay =
dataRow.Field<bool>("AllDay"), Recurrence =
dataRow.Field<bool>("Recurrence"), RecurrenceRule =
dataRow.Field<string>("RecurrenceRule"), Description =
dataRow.Field<string>("Description"), StartTimeZone =
dataRow.Field<string>("StartTimeZone"), EndTimeZone =
dataRow.Field<string>("EndTimeZone") }).ToList();
myAccessConn.Close();
return Json(datasource, JsonRequestBehavior.AllowGet);
}

```

The control code to handle the CRUD operation are as follows.

### C#

```

public JsonResult Batch(EditParams param)
{
    // Mention your own dataSource to be used here.
    string strAccessConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=|DataDirectory|/ScheduleDb.MDB";
    if (param.action == "insert" || (param.action == "batch" && param.added
!= null)) // this block of code will execute while inserting the
appointments
    {
        var value = param.action == "insert" ? param.value : param.added[0];
        using (OleDbConnection myCon = new OleDbConnection(strAccessConn))
        {
            OleDbCommand cmd = new OleDbCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "INSERT INTO
DefaultSchedule (Subject, StartTime, EndTime, AllDay, Recurrence, RecurrenceRul
e, Description, StartTimeZone, EndTimeZone) VALUES
(@Subject, @StartTime, @EndTime, @AllDay, @Recurrence, @RecurrenceRule, @Descri
ption, @StartTimeZone, @EndTimeZone)";
            if (string.IsNullOrEmpty(value.Subject))
                cmd.Parameters.AddWithValue("@Subject", DBNull.Value);
            else
                cmd.Parameters.AddWithValue("@Subject", value.Subject);
            cmd.Parameters.AddWithValue("@StartTime", value.StartTime);
            cmd.Parameters.AddWithValue("@EndTime", value.EndTime);
            cmd.Parameters.AddWithValue("@AllDay", value.AllDay);
            cmd.Parameters.AddWithValue("@Recurrence", value.Recurrence);
            if (string.IsNullOrEmpty(value.RecurrenceRule))
                cmd.Parameters.AddWithValue("@RecurrenceRule", DBNull.Value);
            else
                cmd.Parameters.AddWithValue("@RecurrenceRule", value.RecurrenceRule);
            if (string.IsNullOrEmpty(value.Description))
                cmd.Parameters.AddWithValue("@Description", DBNull.Value);
            else
                cmd.Parameters.AddWithValue("@Description", value.Description);
            if (string.IsNullOrEmpty(value.StartTimeZone))
                cmd.Parameters.AddWithValue("@StartTimeZone", DBNull.Value);

```

```

else
cmd.Parameters.AddWithValue("@StartTimeZone", value.StartTimeZone);
if (string.IsNullOrEmpty(value.EndTimeZone))
cmd.Parameters.AddWithValue("@EndTimeZone", DBNull.Value);
else
cmd.Parameters.AddWithValue("@EndTimeZone", value.EndTimeZone);
cmd.Connection = myCon;
myCon.Open();
cmd.ExecuteNonQuery();
myCon.Close();
}
}
if (param.action == "remove" || param.deleted != null) // this block of
code will execute while removing the appointment
{
if (param.action == "remove")
{
using (OleDbConnection myCon = new OleDbConnection(strAccessConn))
{
myCon.Open();
OleDbCommand cmd = new OleDbCommand("DELETE FROM DefaultSchedule WHERE Id
= @Key", myCon);
cmd.Parameters.AddWithValue("@Key", param.key);
cmd.ExecuteNonQuery();
myCon.Close();
}
}
else
{
foreach (var apps in param.deleted)
{
using (OleDbConnection myCon = new OleDbConnection(strAccessConn))
{
myCon.Open();
OleDbCommand cmd = new OleDbCommand("DELETE FROM DefaultSchedule WHERE Id
= @Key", myCon);
cmd.Parameters.AddWithValue("@Key", apps.Id);
cmd.ExecuteNonQuery();
myCon.Close();
}
}
}
}
if ((param.action == "batch" && param.changed != null) || param.action ==
"update") // this block of code will execute while updating the
appointment
{
var value = param.action == "update" ? param.value : param.changed[0];
using (OleDbConnection myCon = new OleDbConnection(strAccessConn))
{
myCon.Open();
OleDbCommand cmd = new OleDbCommand("UPDATE DefaultSchedule SET
Subject=@Subject,StartTime=@StartTime,EndTime=@EndTime,AllDay=@AllDay,Rec
urrence=@Recurrence,RecurrenceRule=@RecurrenceRule,Description=@Descripti
on,StartTimeZone=@StartTimeZone,EndTimeZone=@EndTimeZone WHERE Id =
@Key", myCon);
if (string.IsNullOrEmpty(value.Subject))

```

```

cmd.Parameters.AddWithValue("@Subject", DBNull.Value);
else
cmd.Parameters.AddWithValue("@Subject", value.Subject);
cmd.Parameters.AddWithValue("@StartTime", value.StartTime);
cmd.Parameters.AddWithValue("@EndTime", value.EndTime);
cmd.Parameters.AddWithValue("@AllDay", value.AllDay);
cmd.Parameters.AddWithValue("@Recurrence", value.Recurrence);
if (string.IsNullOrEmpty(value.RecurrenceRule))
cmd.Parameters.AddWithValue("@RecurrenceRule", DBNull.Value);
else
cmd.Parameters.AddWithValue("@RecurrenceRule", value.RecurrenceRule);
if (string.IsNullOrEmpty(value.Description))
cmd.Parameters.AddWithValue("@Description", DBNull.Value);
else
cmd.Parameters.AddWithValue("@Description", value.Description);
if (string.IsNullOrEmpty(value.StartTimeZone))
cmd.Parameters.AddWithValue("@StartTimeZone", DBNull.Value);
else
cmd.Parameters.AddWithValue("@StartTimeZone", value.StartTimeZone);
if (string.IsNullOrEmpty(value.EndTimeZone))
cmd.Parameters.AddWithValue("@EndTimeZone", DBNull.Value);
else
cmd.Parameters.AddWithValue("@EndTimeZone", value.EndTimeZone);
cmd.Parameters.AddWithValue("@Key", value.Id);
cmd.ExecuteNonQuery();
myCon.Close();
}
}

OleDbConnection myAccessConn = new OleDbConnection(strAccessConn);
OleDbCommand myAccessCommand = new OleDbCommand("SELECT * FROM
DefaultSchedule", myAccessConn);
OleDbDataAdapter myDataAdapter = new OleDbDataAdapter(myAccessCommand);
DataSet myDataSet = new DataSet();
myAccessConn.Open();
myDataAdapter.Fill(myDataSet, "DefaultSchedule");
List<ScheduleData> datasource = new List<ScheduleData>();
datasource = myDataSet.Tables[0].AsEnumerable().Select(dataRow => new
ScheduleData { Id = dataRow.Field<int>("Id"), Subject =
dataRow.Field<string>("Subject"), StartTime =
dataRow.Field<DateTime>("StartTime"), EndTime =
dataRow.Field<DateTime>("EndTime"), AllDay =
dataRow.Field<bool>("AllDay"), Recurrence =
dataRow.Field<bool>("Recurrence"), RecurrenceRule =
dataRow.Field<string>("RecurrenceRule"), Description =
dataRow.Field<string>("Description"), StartTimeZone =
dataRow.Field<string>("StartTimeZone"), EndTimeZone =
dataRow.Field<string>("EndTimeZone") }).ToList();
myAccessConn.Close();
return Json(datasource, JsonRequestBehavior.AllowGet);
}

// Class definition for EditParams to be used as parameter in the above
Crud method for receiving the object value in it.
public class EditParams
{
public string key { get; set; }
public string action { get; set; }
public List<ScheduleData> added { get; set; }
}

```

```

public List<ScheduleData> changed { get; set; }
public List<ScheduleData> deleted { get; set; }
public ScheduleData value { get; set; }
}

```

### ASP.NET Web Method Binding

The Schedule appointment data can retrieve data from ASP.NET Web methods by making use of the UriAdaptor of ejDataManager.

### HTML

```

<!-- HTML element will initialize as a ejSchedule -->
<div id="schedule"></div>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
// get the appointments data from Web method
var dataManager = ej.DataManager({
url: "WebService1.asmx/GetData", // This will trigger to bind the
appointments data to schedule control
batchUrl: "WebService1.asmx/Crud", // This will trigger while saving the
appointment through detail window
insertUrl: "WebService1.asmx/add", // This will trigger while saving the
appointment through quick window
updateUrl: "WebService1.asmx/update", //This will trigger while saving
the resize or drag and drop the appointment
removeUrl: "WebService1.asmx/remove", // This will trigger to delete the
single appointment
adaptor: new ej.WebMethodAdaptor()
});
var sample = new ej.Schedule($("#schedule"), {
currentDate: new Date(2014, 4, 5),
appointmentSettings: {
// Configure the dataSource with dataManager object
dataSource: dataManager
}
});
});
}

```

The server-side code to handle the CRUD operations are as follows.

### C#

```

// To retrieve the appointments and bind it to Scheduler
[WebMethod]
[ScriptMethod(ResponseFormat = ResponseFormat.Json)]
public static object GetData(String CurrentView, String CurrentAction,
DateTime CurrentDate)
{

```

```

// ScheduleAppointmentsObjData is a user-defined class needs to be
defined with collection of scheduler appointments
ScheduleAppointmentsObjDatum obj = new ScheduleAppointmentsObjDatum();
IList<ScheduleAppointmentsObjData> appoint = obj.GetRecords().ToList();
return appoint;
}

public class ScheduleAppointmentsObjDatum
{
    [DataObjectMethod(DataObjectMethodType.Select)]
    public List<ScheduleAppointmentsObjData> GetRecords()
    {
        List<ScheduleAppointmentsObjData> list = new
        List<ScheduleAppointmentsObjData>();
        list.Add(new ScheduleAppointmentsObjData(100, "Bering Sea Gold",
        "chennai", "05/02/2014 09:00:00 AM", "05/02/2014 10:30:00 AM", "", "1",
        "", true, "", "", "", "", false, "", "",
        "FREQ=DAILY;INTERVAL=2;COUNT=10"));
        list.Add(new ScheduleAppointmentsObjData(101, "Bering Sea Gold", "mum",
        "05/02/2014 04:00:00 AM", "05/02/2014 05:00:00 AM", "", "1", "", false,
        "", "", "", "", false, "", "", ""));
        list.Add(new ScheduleAppointmentsObjData(102, "Bering Sea Gold", "trcy",
        "05/02/2014 04:00:00 PM", "05/02/2014 05:30:00 PM", "", "1", "", false,
        "", "", "", "", false, "", "", ""));
        list.Add(new ScheduleAppointmentsObjData(103, "What Happened Next?",
        "chennai", "05/04/2014 03:00:00 AM", "05/04/2014 04:30:00 AM", "", "1",
        "", false, "", "", "", "", false, "", "", ""));
        list.Add(new ScheduleAppointmentsObjData(104, "Bering Sea Gold", "trcy",
        "05/04/2014 05:00:00 AM", "05/04/2014 05:40:00 AM", "", "1", "", false,
        "", "", "", "", false, "", "", ""));
        list.Add(new ScheduleAppointmentsObjData(105, "Daily Planet", "chennai",
        "05/03/2014 01:00:00 AM", "05/03/2014 02:00:00 AM", "", "1", "", false,
        "", "", "", "", false, "", "", ""));
        list.Add(new ScheduleAppointmentsObjData(106, "Alaska: The Last
        Frontier", "chennai", "05/03/2014 08:00:00 AM", "05/03/2014 09:00:00 AM",
        "", "1", "", false, "", "", "", "", false, "", "", ""));
        list.Add(new ScheduleAppointmentsObjData(107, "How It's Made", "chennai",
        "05/01/2014 06:00:00 AM", "05/01/2014 06:30:00 AM", "", "1", "", true,
        "", "", "", "", false, "", "",
        "FREQ=WEEKLY;BYDAY=MO,TU;INTERVAL=1;COUNT=15"));
        list.Add(new ScheduleAppointmentsObjData(108, "Deadest Catch", "chennai",
        "05/03/2014 04:00:00 PM", "05/03/2014 05:00:00 PM", "", "1", "", false,
        "", "", "", "", false, "", "", ""));
        list.Add(new ScheduleAppointmentsObjData(109, "MayDay", "chennai",
        "04/30/2014 06:30:00 AM", "04/30/2014 07:30:00 AM", "", "1", "", false,
        "", "", "", "", false, "", "", ""));
        list.Add(new ScheduleAppointmentsObjData(110, "MoonShiners", "chennai",
        "05/02/2014 02:00:00 AM", "05/02/2014 02:30:00 AM", "", "1", "", true,
        "", "", "", "", false, "", "", "FREQ=DAILY;INTERVAL=1;COUNT=5"));
        list.Add(new ScheduleAppointmentsObjData(111, "Close Encounters",
        "chennai", "04/30/2014 02:00:00 PM", "04/30/2014 03:00:00 PM", "", "1",
        "", true, "", "", "", "", false, "", "",
        "FREQ=WEEKLY;BYDAY=MO,TH;INTERVAL=1;COUNT=5"));
        list.Add(new ScheduleAppointmentsObjData(112, "Close Encounters", "mum",
        "04/30/2014 03:00:00 AM", "04/30/2014 03:30:00 AM", "", "1", "", true,
        "", "", "", "", false, "", "",
        "FREQ=WEEKLY;BYDAY=WE;INTERVAL=1;COUNT=3"));
    }
}

```

```

list.Add(new ScheduleAppointmentsObjData(113, "Highway Through Hell",
"chennai", "05/01/2014 03:00:00 AM", "05/01/2014 07:00:00 AM", "", "1",
"", true, "", "", "", "", false, "", "",
"FREQ=DAILY;INTERVAL=2;COUNT=10"));
list.Add(new ScheduleAppointmentsObjData(114, "Moon Shiners", "chennai",
"05/02/2014 04:20:00 AM", "05/02/2014 05:50:00 AM", "", "1", "", false,
"", "", "", "", false, "", "", ""));
list.Add(new ScheduleAppointmentsObjData(115, "Cash Cab", "chennai",
"04/30/2014 03:00:00 PM", "04/30/2014 04:30:00 PM", "", "1", "", true,
"", "", "", "", false, "", "", "FREQ=DAILY;INTERVAL=1;COUNT=5"));
return list;
}
}
[Serializable]
public class ScheduleAppointmentsObjData
{
private int _id;
private String _subject;
private String _location;
private String _startTime;
private String _endTime;
private String _description;
private String _owner;
private String _priority;
private Boolean _recurrence;
private String _recurrenceType;
private String _recurrenceTypeCount;
private String _remainderCategorize;
private String _customStyle;
private Boolean _allDay;
private String _recurrenceStartDate;
private String _recurrenceEndDate;
private String _recurrenceRule;
public ScheduleAppointmentsObjData()
{
}
public ScheduleAppointmentsObjData(int _id, string _subject, string
_location, string _startTime, string _endTime, string _description,
string _owner, string _priority, bool _recurrence, string
_recurrenceType, string _recurrenceTypeCount, string
_remainderCategorize, string _customStyle, bool _allDay, string
_recurrenceStartDate, string _recurrenceEndDate, string _recurrenceRule)
{
this._id = _id;
this._subject = _subject;
this._location = _location;
this._startTime = _startTime;
this._endTime = _endTime;
this._description = _description;
this._owner = _owner;
this._priority = _priority;
this._recurrence = _recurrence; ;
this._recurrenceType = _recurrenceType;
this._recurrenceTypeCount = _recurrenceTypeCount;
this._remainderCategorize = _remainderCategorize;
this._customStyle = _customStyle;
this._allDay = _allDay;

```

```
this._recurrenceStartDate = _recurrenceStartDate;
this._recurrenceEndDate = _recurrenceEndDate;
this._recurrenceRule = _recurrenceRule;
}
public int ID
{
    get
    {
        return _id;
    }
    set
    {
        _id = value;
    }
}
public string Subject
{
    get
    {
        return _subject;
    }
    set
    {
        _subject = value;
    }
}
public string Location
{
    get
    {
        return _location;
    }
    set
    {
        _location = value;
    }
}
public string StartTime
{
    get
    {
        return _startTime;
    }
    set
    {
        _startTime = value;
    }
}
public string EndTime
{
    get
    {
        return _endTime;
    }
    set
    {
        _endTime = value;
    }
}
```

```
}  
}  
public string Description  
{  
    get  
    {  
        return _description;  
    }  
    set  
    {  
        _description = value;  
    }  
}  
public string Owner  
{  
    get  
    {  
        return _owner;  
    }  
    set  
    {  
        _owner = value;  
    }  
}  
public string Priority  
{  
    get  
    {  
        return _priority;  
    }  
    set  
    {  
        _priority = value;  
    }  
}  
public Boolean Recurrence  
{  
    get  
    {  
        return _recurrence;  
    }  
    set  
    {  
        _recurrence = value;  
    }  
}  
public string RecurrenceType  
{  
    get  
    {  
        return _recurrenceType;  
    }  
    set  
    {  
        _recurrenceType = value;  
    }  
}
```



```
public string RecurrenceTypeCount
{
    get
    {
        return _recurrenceTypeCount;
    }
    set
    {
        _recurrenceTypeCount = value;
    }
}
public string RemainderCategorize
{
    get
    {
        return _remainderCategorize;
    }
    set
    {
        _remainderCategorize = value;
    }
}
public string CustomStyle
{
    get
    {
        return _customStyle;
    }
    set
    {
        _customStyle = value;
    }
}
public Boolean AllDay
{
    get
    {
        return _allDay;
    }
    set
    {
        _allDay = value;
    }
}
public string RecurrenceStartDate
{
    get
    {
        return _recurrenceStartDate;
    }
    set
    {
        _recurrenceStartDate = value;
    }
}
public string RecurrenceEndDate
{

```

```

get
{
    return _recurrenceEndDate;
}
set
{
    _recurrenceEndDate = value;
}
}
public string RecurrenceRule
{
    get
    {
        return _recurrenceRule;
    }
    set
    {
        _recurrenceRule = value;
    }
}
}
// Method to retrieve appointments from the ScheduleAppointmentsObjDatum Class
public static IList<ScheduleAppointmentsObjData> GetAllRecords()
{
    ScheduleAppointmentsObjDatum obj = new ScheduleAppointmentsObjDatum();
    IList<ScheduleAppointmentsObjData> appoint = obj.GetRecords().ToList();
    return appoint;
}
// Method to convert appointment object of dictionary type into valid format.
public static ScheduleAppointmentsObjData
GetObjectValue(Dictionary<string, object> objValue)
{
    Dictionary<string, object> KeyVal = objValue;
    ScheduleAppointmentsObjData appointValue = new
    ScheduleAppointmentsObjData();
    foreach (KeyValuePair<string, object> keyValue in KeyVal)
    {
        if (keyValue.Key == "ID")
            appointValue.ID = Convert.ToInt32(keyValue.Value);
        else if (keyValue.Key == "Subject")
            appointValue.Subject = Convert.ToString(keyValue.Value);
        else if (keyValue.Key == "Location")
            appointValue.Location = Convert.ToString(keyValue.Value);
        else if (keyValue.Key == "StartTime")
            appointValue.StartTime =
            Convert.ToDateTime(keyValue.Value).ToString("MM '/' dd '/' yyyy hh:mm:ss
            tt");
        else if (keyValue.Key == "EndTime")
            appointValue.EndTime =
            Convert.ToDateTime(keyValue.Value).ToString("MM '/' dd '/' yyyy hh:mm:ss
            tt");
        else if (keyValue.Key == "Description")
            appointValue.Description = Convert.ToString(keyValue.Value);
        else if (keyValue.Key == "AllDay")
            appointValue.AllDay = Convert.ToBoolean(keyValue.Value);
    }
}

```

```

else if (keyValue.Key == "Recurrence")
appointValue.Recurrence = Convert.ToBoolean(keyValue.Value);
else if (keyValue.Key == "RecurrenceRule")
appointValue.RecurrenceRule = Convert.ToString(keyValue.Value);
}
return appointValue;
}
// Triggers while creating appointment through quick window.
[WebMethod]
[ScriptMethod(ResponseFormat = ResponseFormat.Json)]
public static void add(object value)
{
ScheduleAppointmentsObjData appointValue = GetObjectValue(value as
Dictionary<string, object>);
GetAllRecords().Insert(0, appointValue);
}
// Triggers while editing the appointments.
[WebMethod]
[ScriptMethod(ResponseFormat = ResponseFormat.Json)]
public static void update(object value)
{
ScheduleAppointmentsObjData obj = GetObjectValue(value as
Dictionary<string, object>);
var filterData = GetAllRecords().ToList().Where(c => c.ID ==
Convert.ToInt32(obj.ID));
if (filterData.Count() > 0)
{
ScheduleAppointmentsObjData appoint = GetAllRecords().Single(A => A.ID ==
Convert.ToInt32(obj.ID));
appoint.StartTime = obj.StartTime;
appoint.EndTime = obj.EndTime;
appoint.Subject = obj.Subject;
appoint.Recurrence = obj.Recurrence;
appoint.AllDay = obj.AllDay;
appoint.RecurrenceRule = obj.RecurrenceRule;
}
}
// Triggers on deleting an appointment.
[WebMethod]
[ScriptMethod(ResponseFormat = ResponseFormat.Json)]
public static void remove(int key)
{
ScheduleAppointmentsObjData removeApp = GetAllRecords().Where(c => c.ID
== key).FirstOrDefault();
if (removeApp != null)
GetAllRecords().Remove(removeApp);
}
// Triggers for all CRUD actions on Scheduler appointments.
[WebMethod]
[ScriptMethod(ResponseFormat = ResponseFormat.Json)]
public static void Crud(List<object> added, List<object> changed,
List<object> deleted)
{
if (added != null && added.Count > 0)
{
ScheduleAppointmentsObjData appointValue = GetObjectValue(added[0] as
Dictionary<string, object>);

```

```

GetAllRecords().Insert(0, appointValue);
}
if (changed != null && changed.Count > 0)
{
    ScheduleAppointmentsObjData value = GetObjectValue(changed[0] as
    Dictionary<string, object>);
    var filterData = GetAllRecords().ToList().Where(c => c.ID ==
    Convert.ToInt32(value.ID));
    if (filterData.Count() > 0)
    {
        ScheduleAppointmentsObjData appoint = GetAllRecords().Where(A => A.ID ==
        Convert.ToInt32(value.ID)).FirstOrDefault();
        appoint.StartTime = value.StartTime;
        appoint.EndTime = value.EndTime;
        appoint.Subject = value.Subject;
        appoint.Recurrence = value.Recurrence;
        appoint.AllDay = value.AllDay;
        appoint.RecurrenceRule = value.RecurrenceRule;
    }
}
if (deleted != null && deleted.Count > 0)
{
    foreach (var delete in deleted)
    {
        ScheduleAppointmentsObjData value = GetObjectValue(delete as
        Dictionary<string, object>);
        ScheduleAppointmentsObjData removeApp = GetAllRecords().Where(c => c.ID
        == value.ID).FirstOrDefault();
        if (removeApp != null)
            GetAllRecords().Remove(removeApp);
    }
}
}

```

### MVC Controller Action Binding

The Schedule appointment data can retrieve data from MVC controller. This can be achieved by using the [UrlAdaptor](#) of [ej.DataManager](#).

### HTML

```

<!-- HTML element will initialize as a ejSchedule -->
<div id="schedule"></div>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
    $(function () {
        /// get the appointments data from Web method
        var dataManager = ej.DataManager({
            url: "Home/GetData", // This will trigger to bind the appointments data
            to schedule control
            batchUrl: "Home/Crud", // This will trigger while saving the appointment
            through detail window
        });
    });
}

```

```

insertUrl: "Home/add", // This will trigger while saving the appointment
through quick window
updateUrl: "Home/update", //This will trigger while saving the resize or
drag and drop the appointment
removeUrl: "Home/remove", // This will trigger to delete the single
appointment
adaptor: new ej.UrlAdaptor()
});
var sample = new ej.Schedule($("#schedule"), {
currentDate: new Date(2014, 4, 5),
appointmentSettings: {
// Configure the dataSource with dataManager object
dataSource: dataManager
}
});
});
}

```

The server-side controller code to handle the CRUD operations are as follows.

### C#

```

// To initially bind the appointments with Scheduler
public JsonResult GetData()
{
// ScheduleDataDataContext is a LINQ-to-SQL data class name that is
defined in the .dbml file to access the tables from the database
IEnumerable data = new ScheduleDataDataContext().Appointments.Take(100);
return Json(data, JsonRequestBehavior.AllowGet);
}
// Triggers while saving a new appointment through quick window
public JsonResult add(Appointment value)
{
ScheduleDataDataContext db = new ScheduleDataDataContext();
int intMax = db.Appointments.ToList().Count > 0 ?
db.Appointments.ToList().Max(p => p.Id) : 1;
Appointment appoint = new Appointment()
{
Id = intMax + 1,
StartTime = value.StartTime,
EndTime = value.EndTime,
Subject = value.Subject,
Description = value.Description,
OwnerId = value.OwnerId,
Recurrence = value.Recurrence,
AllDay = value.AllDay,
RecurrenceRule = value.RecurrenceRule
};
db.Appointments.InsertOnSubmit(appoint);
db.SubmitChanges();
IEnumerable data = new ScheduleDataDataContext().Appointments.Take(100);
return Json(data, JsonRequestBehavior.AllowGet);
}
// Triggers while editing/dragging/resizing the existing appointment
public JsonResult update(Appointment value)
{

```

```

ScheduleDataDataContext db = new ScheduleDataDataContext();
var filterData = db.Appointments.Where(c => c.Id ==
Convert.ToInt32(value.Id));
Appointment appoint = db.Appointments.Single(A => A.Id ==
Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
    DateTime startTime = Convert.ToDateTime(value.StartTime);
    DateTime endTime = Convert.ToDateTime(value.EndTime);
    appoint.StartTime = startTime;
    appoint.EndTime = endTime;
    appoint.Subject = value.Subject;
    appoint.Description = value.Description;
    appoint.OwnerId = value.OwnerId;
    appoint.Recurrence = Convert.ToByte(value.Recurrence);
    appoint.AllDay = value.AllDay;
    appoint.RecurrenceRule = value.RecurrenceRule;
}
db.SubmitChanges();
IEnumerable data = new ScheduleDataDataContext().Appointments.Take(100);
return Json(data, JsonRequestBehavior.AllowGet);
}
// Triggers when an appointment is deleted
public JsonResult remove(string key)
{
    ScheduleDataDataContext db = new ScheduleDataDataContext();
    Appointment app = db.Appointments.Where(c => c.Id ==
Convert.ToInt32(key)).FirstOrDefault();
    if (app != null) db.Appointments.DeleteOnSubmit(app);
    db.SubmitChanges();
    IEnumerable data = new ScheduleDataDataContext().Appointments.Take(100);
    return Json(data, JsonRequestBehavior.AllowGet);
}
// Triggers for any of the Scheduler CRUD operation
public JsonResult Crud(EditParams param)
{
    ScheduleDataDataContext db = new ScheduleDataDataContext();
    if (param.action == "insert" || (param.action == "batch" && param.added
!= null)) // this block of code will execute while inserting the
appointments
    {
        var value = param.action == "insert" ? param.value : param.added[0];
        int intMax = db.Appointments.ToList().Count > 0 ?
db.Appointments.ToList().Max(p => p.Id) : 1;
        DateTime startTime = Convert.ToDateTime(value.StartTime);
        DateTime endTime = Convert.ToDateTime(value.EndTime);
        Appointment appoint = new Appointment()
        {
            Id = intMax + 1,
            StartTime = startTime,
            EndTime = endTime,
            Subject = value.Subject,
            Description = value.Description,
            OwnerId = value.OwnerId,
            Recurrence = value.Recurrence,
            AllDay = value.AllDay,
            RecurrenceRule = value.RecurrenceRule
        }
    }
}

```

```

};
db.Appointments.InsertOnSubmit(appoint);
db.SubmitChanges();
}
else if (param.action == "remove") // this block of code will execute
while removing the appointment
{
Appointment app = db.Appointments.Where(c => c.Id ==
Convert.ToInt32(param.key)).FirstOrDefault();
if (app != null) db.Appointments.DeleteOnSubmit(app);
db.SubmitChanges();
}
else if ((param.action == "batch" && param.changed != null) ||
param.action == "update") // this block of code will execute while
updating the appointment
{
var value = param.action == "update" ? param.value : param.changed[0];
var filterData = db.Appointments.Where(c => c.Id ==
Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
Appointment appoint = db.Appointments.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appoint.StartTime = startTime.ToUniversalTime();
appoint.EndTime = endTime.ToUniversalTime();
appoint.Subject = value.Subject;
appoint.Description = value.Description;
appoint.OwnerId = value.OwnerId;
appoint.Recurrence = Convert.ToByte(value.Recurrence);
appoint.AllDay = value.AllDay;
appoint.RecurrenceRule = value.RecurrenceRule;
}
db.SubmitChanges();
}
IEnumerable data = new ScheduleDataDataContext().Appointments.Take(500);
return Json(data, JsonRequestBehavior.AllowGet);
}
// Class definition for EditParams to be used as parameter in the above
Crud method for receiving the object value in it.
public class EditParams
{
public string key { get; set; }
public string action { get; set; }
public List<Appointment> added { get; set; }
public List<Appointment> changed { get; set; }
public Appointment value { get; set; }
}

```

### Loading Data on Demand

Load on demand feature allows the Scheduler to retrieve only the filtered appointment data (for the current Scheduler date range) from the service/database during **loading time**, and that too only for the current Scheduler view. There are 3 parameters made available on the server-side namely **CurrentDate**, **CurrentView** and **CurrentAction** through which only the necessary appointments are retrieved from the

database and then assigned to the Scheduler dataSource. With this kind of Scheduler action, consuming only lesser data will reduce the usage of network bandwidth size and loading time.

The **enableLoadOnDemand** property is used to enable or disable the load on demand functionality of the schedule.

### HTML

```
<!-- HTML element will initialize as a ejSchedule -->
<div id="schedule"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
/// get the appointments data from Web method
var dataManager = ej.DataManager({
url: "Home/GetData", // This will trigger to bind the appointments data
to schedule control
batchUrl: "Home/Crud", // This will trigger while saving the appointment
through detail window
insertUrl: "Home/add", // This will trigger while saving the appointment
through quick window
updateUrl: "Home/update", //This will trigger while saving the resize or
drag and drop the appointment
removeUrl: "Home/remove", // This will trigger to delete the single
appointment
adaptor: new ej.UrlAdaptor()
});
var sample = new ej.Schedule($("#schedule"), {
currentDate: new Date(2014, 4, 5),
appointmentSettings: {
// Configure the dataSource with dataManager object
dataSource: dataManager
}
});
});
});
}
```

The server-side code to handle the load on demand is as follows.

### C#

```
// retrieve the appointments based on the current date.
public IEnumerable<Event> GetData(String CurrentDate, String CurrentView,
String CurrentAction)
{
var dateString = Regex.Match(CurrentDate.ToString(),
@"^(\\w+\\b.*?){4}").ToString();
string format = "ddd MMM dd yyyy";
DateTime dateTimeValue;
try
{

```



```

dateTimeValue = DateTime.ParseExact(dateString, format,
CultureInfo.InvariantCulture);
}
catch (FormatException)
{
    var dateSplit = CurrentDate.Split(' '); //For IE<=10 Fri Mar 20 11:00:22
UTC+0530 2015
    if (dateSplit[2].Length == 1) dateSplit[2] = string.Concat("0",
dateSplit[2]);
    dateString = string.Concat(dateSplit[0], ' ', dateSplit[1], ' ',
dateSplit[2], ' ', dateSplit[dateSplit.Length - 1]);
    dateTimeValue = DateTime.ParseExact(dateString, format,
CultureInfo.InvariantCulture);
}
// AppointmentDeposit is a user-defined class within which the
FilterAppointment method is defined.
AppointmentDeposit rep = new AppointmentDeposit();
var data = rep.FilterAppointment(dateTimeValue, CurrentAction,
CurrentView);
return data;
}
// Method to filter the appointments based on the date range
public List<Event> FilterAppointment(DateTime CurrentDate, String
CurrentAction, String CurrentView)
{
    DateTime CurrDate = Convert.ToDateTime(CurrentDate);
    DateTime StartDate = FirstWeekDate(CurrDate.Date);
    DateTime EndDate = FirstWeekDate(CurrDate.Date);
    List<Event> appointmentList = new NORTHWNDEntities().Events.ToList();
    switch (CurrentView)
    {
        case "day":
            StartDate = CurrentDate;
            EndDate = CurrentDate;
            break;
        case "week":
            EndDate = EndDate.AddDays(7);
            break;
        case "workweek":
            EndDate = EndDate.AddDays(5);
            break;
        case "month":
            StartDate = CurrDate.Date.AddDays(-CurrDate.Day + 1);
            EndDate = StartDate.AddMonths(1);
            break;
    }
    appointmentList = new NORTHWNDEntities().Events.ToList().Where(app =>
((Convert.ToDateTime(app.StartTime).Date >=
Convert.ToDateTime(StartDate.Date)) &&
(Convert.ToDateTime(app.EndTime).Date <=
Convert.ToDateTime(EndDate.Date)))).ToList();
    return appointmentList;
}
internal static DateTime FirstWeekDate(DateTime CurrentDate)
{
    try
    {

```

```
DateTime FirstDayOfWeek = CurrentDate;
DayOfWeek WeekDay = FirstDayOfWeek.DayOfWeek;
switch (WeekDay)
{
    case DayOfWeek.Sunday:
        break;
    case DayOfWeek.Monday:
        FirstDayOfWeek = FirstDayOfWeek.AddDays(-1);
        break;
    case DayOfWeek.Tuesday:
        FirstDayOfWeek = FirstDayOfWeek.AddDays(-2);
        break;
    case DayOfWeek.Wednesday:
        FirstDayOfWeek = FirstDayOfWeek.AddDays(-3);
        break;
    case DayOfWeek.Thursday:
        FirstDayOfWeek = FirstDayOfWeek.AddDays(-4);
        break;
    case DayOfWeek.Friday:
        FirstDayOfWeek = FirstDayOfWeek.AddDays(-5);
        break;
    case DayOfWeek.Saturday:
        FirstDayOfWeek = FirstDayOfWeek.AddDays(-6);
        break;
}
return (FirstDayOfWeek);
}
catch
{
    return DateTime.Now;
}
```

## Views

The number of days and its associated appointments are usually grouped together in Scheduler to organize different views. The available view options in Scheduler are as follows,

- Day
- Week
- Workweek
- Month
- Custom View
- Agenda
- Timeline View

Usually these view options are displayed as a toolbar in the date-header section of the Schedule control. The items within the views toolbar can be added/removed based on the value passed to the [views](#) property.

By default, the Schedule control's active view is **Week** view. Also, it is possible to change the active view of the Scheduler by setting [currentView](#) option with the required view name as depicted below.

## HTML

```
<!--Container for ejScheduler widget-->
<div id="schedule"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
//Required views display the control
views: ["Day", "WorkWeek"],
// Set the Active view
currentView: ej.Schedule.CurrentView.Workweek
});
});
}
```

**Note:** The **currentView** property accepts both the string and **ej.Schedule.CurrentView** enum value.

#### Day

It represents a single day Scheduler view (single date display) with all its related appointments.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="schedule"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
// Set the Active view
currentView: ej.Schedule.CurrentView.Day,
currentDate: new Date(2015, 11, 7),
appointmentSettings: {
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 06:00 AM"),
EndTime: new Date("2015/11/7 07:00 AM")
}, {
Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 9:00 AM"),
EndTime: new Date("2015/11/7 02:30 PM")
}]
}
});
});
}
```

## Week

It's a view displaying a count of 7 days (from Sunday to Saturday) with all its related appointments. The first day of the week can be changed using the [firstDayOfWeek](#) API which accepts either the `integer` (Sunday=0, Monday=1, Tuesday=2, etc) or `string` ("Sunday", "Monday", etc) or `ej.Schedule.DayOfWeek` enum type value. The default value of this **firstDayOfWeek** depends on the current culture (language) used in the Scheduler.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="schedule"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
// Set the Active view
currentView: ej.Schedule.CurrentView.Week,
// Configure the week start day(First day of week)
firstDayOfWeek: ej.Schedule.FirstDayOfWeek.Monday,
currentDate: new Date(2015, 11, 7),
appointmentSettings: {
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 06:00 AM"),
EndTime: new Date("2015/11/7 07:00 AM")
}, {
Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 9:00 AM"),
EndTime: new Date("2015/11/7 02:30 PM")
}]
}
});
});
}
```

## Work Week

Work week view displays the working days of the week (count of 5 days) and its associated appointments. It is also possible to customize the days to be displayed in the work week view using [workWeek](#) API which accepts the string array such as ["Monday", "Tuesday", "Wednesday", "Thursday" and "Friday"]. By default, it renders from Monday to Friday (5 days).

### HTML

```
<!--Container for ejScheduler widget-->
<div id="schedule"></div>
```

**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
// Set the Active view
currentView: ej.Schedule.CurrentView.Workweek,
// configure the work week days
workWeek: ["Monday", "Tuesday", "Thursday", "Friday", "Saturday"],
currentDate: new Date(2015, 11, 7),
appointmentSettings: {
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 06:00 AM"),
EndTime: new Date("2015/11/7 07:00 AM")
}, {
Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 9:00 AM"),
EndTime: new Date("2015/11/7 02:30 PM")
}]
}
});
});
}

```

**Month**

Month view displays the entire days of a particular month and all its related appointments. An alternative way to navigate to a particular date in a day view directly from Month view, clicking on the appropriate month cell date header will do so. If the week date range column is clicked, it will navigate to the corresponding week view.

The next and previous month date cells in the Month view can be shown/hidden on the Scheduler using [showNextPrevMonth](#) property by setting it to *false*.

For example – To set the Month view as current view in Scheduler and to hide the other month days in it, refer the below code example.

**HTML**

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {

```

```
// Set the Active view as Month
currentView: ej.Schedule.CurrentView.Month,
showNextPrevMonth: false,
currentDate: new Date(2015, 11, 7),
appointmentSettings: {
  //Array of JSON data configure in dataSource
  dataSource: [{
    Id: 1,
    Subject: "Music Class",
    StartTime: new Date("2015/11/7 06:00 AM"),
    EndTime: new Date("2015/11/7 07:00 AM")
  }, {
    Id: 2,
    Subject: "School",
    StartTime: new Date("2015/11/7 9:00 AM"),
    EndTime: new Date("2015/11/7 02:30 PM")
  }]
}
});
});
}
```

---

**Note:** An appointment directly created in Month view will be considered as an all-day appointment.

---

### Custom

The Scheduler can be displayed with the user-specified date ranges, such as 4 days or any specific date ranges instead of default view options, by making use of the [renderDates](#) property. This property includes two sub properties namely **start** and **end**, which accepts the date object or date value in string format to specify the date range.

To display the custom view option in the toolbar-like view options in the scheduler header area, add the **CustomView** value to the views property array collection as shown below.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="schedule"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#schedule"), {
      // We can add the "CustomView" in views collection
      views: ["Day", "Week", "WorkWeek", "Month", "CustomView"],
      currentDate: new Date(2015, 11, 6),
      // Configure the custom date
      renderDates: {
        // Render start date
        start: new Date(2015, 11, 6),
        // Render end date
        end: new Date(2015, 11, 9)
      },
    },
```

```
// Set the Active view
currentView: ej.Schedule.CurrentView.CustomView,
appointmentSettings: {
  //Array of JSON data configure in dataSource
  dataSource: [{
    Id: 1,
    Subject: "Music Class",
    StartTime: new Date("2015/11/7 06:00 AM"),
    EndTime: new Date("2015/11/7 07:00 AM")
  }, {
    Id: 2,
    Subject: "School",
    StartTime: new Date("2015/11/7 9:00 AM"),
    EndTime: new Date("2015/11/7 02:30 PM")
  }]
}
});
});
}
```

When the date difference between the provided start and end date is greater than 7, then the month-like view will get displayed in Vertical Scheduler mode - whereas with the date difference less than 7 days displays the Scheduler with exact count of the specified days.

**Note:** When the `currentDate` property of Scheduler is set with a date, that lies beyond the specified custom date range - then the Scheduler navigates to the current date with the mentioned date differences.

### Agenda

This View option lists out the appointments in a grid-like view for the next 7 days by default from the current date. The count of the number of appointments to be listed in this view can be customized using the [agendaViewSettings.daysInAgenda](#).

### HTML

```
<!--Container for ejScheduler widget-->
<div id="schedule"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#schedule"), {
      // Set the Active view
      currentView: ej.Schedule.CurrentView.Agenda,
      currentDate: new Date(2015, 11, 7),
      //configure the agenda view
      agendaViewSettings: {
        //Next 5 days Appointments lists out from current date
        daysInAgenda: 5
      },
      appointmentSettings: {
        //Array of JSON data configure in dataSource

```

```

dataSource: [{
  Id: 1,
  Subject: "Music Class",
  StartTime: new Date("2015/11/7 06:00 AM"),
  EndTime: new Date("2015/11/7 07:00 AM")
}, {
  Id: 2,
  Subject: "School",
  StartTime: new Date("2015/11/7 9:00 AM"),
  EndTime: new Date("2015/11/7 02:30 PM")
}]
}
});
});
}

```

**Note:** In Agenda view, the templates can be applied for the date and time columns which can be referred [here](#). Also, the template passed through the [appointmentTemplateID](#) will get applied to the event column in Agenda view.

#### Restriction on View Navigation

It is possible to restrict the users to display only the specific list of views in the Schedule header section and also not to navigate to other views that are not listed.

**For example,** if the views property is set only with Month view – then the Schedule header section displays only the Month option in the view toolbar and also other additional available actions like navigating to day/week view on clicking the month header dates and week date-range is stopped.

#### HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

#### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#schedule"), {
      // Add only the "Month" in views collection
      views: ["Month"],
      currentDate: new Date(2015, 11, 7),
      appointmentSettings: {
        //Array of JSON data configure in dataSource
        dataSource: [{
          Id: 1,
          Subject: "Music Class",
          StartTime: new Date("2015/11/7 06:00 AM"),
          EndTime: new Date("2015/11/7 07:00 AM")
        }, {
          Id: 2,
          Subject: "School",
          StartTime: new Date("2015/11/7 9:00 AM"),
          EndTime: new Date("2015/11/7 02:30 PM")
        }
      ]
    });
  });
}

```



```

    }]
  }
});
});
}

```

**Note:** Even though Week view is the active view in Scheduler by default, if it is not listed in the views collection – then the first listed option in the views collection will be taken as current view of the Scheduler.

### Timeline View

Timeline view displays the day, time and its associated events horizontally arranged from left to right. By default, Scheduler renders in vertical mode and it can be changed to the timeline mode using [orientation](#) property which accepts both the [string](#) and [ej.Schedule.Orientation](#) enum value.

All the applicable features in Vertical mode works similar with Timeline mode (Horizontal) and only the visualization of the layout changes based on the orientation.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#schedule"), {
      currentDate: new Date(2015, 11, 7),
      //set the timeline (horizontal) view
      orientation: ej.Schedule.Orientation.Horizontal,
      appointmentSettings: {
        //Array of JSON data configure in dataSource
        dataSource: [{
          Id: 1,
          Subject: "Music Class",
          StartTime: new Date("2015/11/7 09:00 AM"),
          EndTime: new Date("2015/11/7 10:00 AM")
        }, {
          Id: 2,
          Subject: "School",
          StartTime: new Date("2015/11/7 02:00 PM"),
          EndTime: new Date("2015/11/7 06:30 PM")
        }
      ]
    });
  });
}

```

### Working with Appointments

An appointment represents a certain time interval in a schedule cell depicting a plan made for the specified time interval.

## Appointment Types

The types of appointments available within Scheduler can be categorized as follows

### Normal

Represents an appointment that is created for a certain time interval in one or more number of days. If the normal appointment is created for more than 24 hours, then those longer appointments will be rendered on the all-day row.

---

**Note:** If the normal appointment is to be created for two days (say from November 25, 2015 – 11.00 PM to November 26, 2015 2.00 AM) but less than 24 hour time interval, then the appointment is split into two partitions and will be displayed appropriately on both the days.

---

### All-Day

Represents an appointment that is created for an entire day such as holiday events. It renders separately in an All-day row, a separate place for all-day appointments. In Timeline (horizontal) view, all-day appointment renders in the usual work cells, as no all-day cells are present in that view.

---

**Note:** An all-day row is normally visible on the Scheduler, as the [showAllDayRow](#) property is set to true by default.

---

### Recurrence

Represents an appointment that is created for a certain time interval that occurs repeatedly in a daily, weekly, monthly, yearly or every weekday basis at the same time interval based on the recurrence rule. The other available options and validations that can be performed on recurrence appointments can be referred from [here](#).

## CRUD operation

Appointments play a dynamic role within the Schedule control with which the users mostly interact. You can manipulate (add/edit/delete/drag/resize) the required appointments that reveals one of the main purpose of the Schedule control.

### Add/Edit Appointments

The appointments can be added/edited in the Scheduler using any one of the following ways,

- Quick window
- Inline creation/editing
- Default appointment window
- [Context menu](#)
- Through programmatically

### Quick Window

The Quick window usually pops out while single clicking on the Scheduler cells or appointments. It requires the user to enter the Subject to proceed with the appointment creation. It also includes an **Edit Appointment** option displayed at the bottom left corner – on selection which opens up the normal appointment window.

On single clicking the scheduler appointments, the pop-up that shows up contains the appointment information along with the other options that are listed below,

- Edit Appointment
- Edit Series (only for the recurrence appointments)
- Delete icon

The quick window option can be enabled/disabled by using [showQuickWindow](#) API, whereas its default value is set to **true**.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="schedule"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
//disables the quick window
showQuickWindow: false,
currentDate: new Date(2015, 11, 7),
appointmentSettings: {
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 06:00 AM"),
EndTime: new Date("2015/11/7 07:00 AM")
}, {
Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 9:00 AM"),
EndTime: new Date("2015/11/7 02:30 PM")
}]
}
});
});
}
```

**Note:** Select multiple cells either using mouse or keyboard access keys (shift+arrow keys) and press enter key, so that the quick window opens up for the selected date/time range.

Another way to disable the quick window option at dynamic time can be achieved through the [cellClick](#) and [appointmentClick](#) events. The below code example shows the way to disable the quick appointment window only while clicking on the cells, but displays for appointments.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="schedule"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
```

```

currentDate: new Date(2015, 11, 7),
showQuickWindow: true,
appointmentSettings: {
  //Array of JSON data configure in dataSource
  dataSource: [{
    Id: 1,
    Subject: "Music Class",
    StartTime: new Date("2015/11/7 06:00 AM"),
    EndTime: new Date("2015/11/7 07:00 AM")
  }, {
    Id: 2,
    Subject: "School",
    StartTime: new Date("2015/11/7 9:00 AM"),
    EndTime: new Date("2015/11/7 02:30 PM")
  }]
},
cellClick: "onCellClick"
});
});
}

```

## JAVASCRIPT

```

function onCellClick(args) {
  args.cancel = true; // Prevents the display of quick window on clicking
  the cells.
}

```

## Inline Appointment Creation/Editing

Another easier way, for adding or editing the appointment's subject alone can be achieved using inline Add/Edit support. It allows the user to add and edit the appointments inline.

To get familiar with inline Add mode, single click on any of the Scheduler cells or press **enter** key on the selected cells. When the inline adding mode is ON, a text box will get created within the clicked Scheduler cells with a blinking cursor in it, requiring the user to enter the subject of an appointment. Once the subject is entered, the appointment will be saved on pressing the **enter** key.

To enable inline edit mode, single click on any of the existing appointment's subject, so that the user can edit the subject of that appointment. The edited subject of that appointment is then updated on pressing the **enter** key.

The inline option can be enabled/disabled on Scheduler by using the [allowInline](#) API, whereas its default value is set to **false**.

## HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

## JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {

```

```

$(function () {
var sample = new ej.Schedule($("#schedule"), {
//enables the inline adding/editing on Scheduler
allowInline: true,
currentDate: new Date(2015, 11, 7),
appointmentSettings: {
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 06:00 AM"),
EndTime: new Date("2015/11/7 07:00 AM")
}, {
Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 9:00 AM"),
EndTime: new Date("2015/11/7 02:30 PM")
}]
}
});
});
}

```

#### Enabling Inline Edit alone

It is possible to disable the inline appointment creation and enabling only the editing mode of inline by making use of the `cellClick` event. The below code example shows the way to disable the inline appointment creation while clicking on the cells, but appointments can be edited while clicking on it's subject.

#### HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

#### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
currentDate: new Date(2015, 11, 7),
showQuickWindow: true,
appointmentSettings: {
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 06:00 AM"),
EndTime: new Date("2015/11/7 07:00 AM")
}, {
Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 9:00 AM"),
EndTime: new Date("2015/11/7 02:30 PM")
}
]
}
});
});
}

```

```
    }]  
    },  
    cellClick: "onCellClick"  
  });  
});  
}
```

### JAVASCRIPT

```
function onCellClick(args) {  
  args.cancel = true; // Prevents inline appointment creation on clicking  
  the cells.  
}
```

### Default Appointment Window

The default appointment window is availed with options like

- Subject
- Start and End Time
- All-Day
- TimeZone (for both Start and End time)
- Repeat options
- Description

The other additional options available are listed below for which the appropriate API's are needed to be configured to display these options on the appointment window.

- Location ([showLocationField](#))
- Priority ([prioritySettings](#))
- Categorize ([categorizeSettings](#))
- [Resources](#)

The appointments can be created by double-clicking the Scheduler cells across the required time slots, which makes the Create Appointment window to pop-up. The start and end time fields will get automatically populated, according to the time-slot selection. Clicking on the done button in an appointment window will create the appointment for the selected time cells.

**Note:** Select multiple cells both using mouse or keyboard access keys (shift+arrow keys) and press Alt+N key, so that the default appointment window opens up for the selected date/time range with the Start and End time fields automatically filled in.

To prevent the display of default appointment window on double clicking the Scheduler cells, either the [appointmentWindowOpen](#) or [cellDoubleClick](#) event can be used, within which the **args.cancel** needs to be set to true. This behavior is depicted in the below code example.

### HTML

```
<!--Container for ejScheduler widget-->  
<div id="Schedule1"></div>
```

**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
width: "100%",
height: "525px",
currentDate: new Date(2015, 11, 5),
appointmentSettings: {
dataSource: [{
Id: 101,
Subject: "Talk with Nature",
StartTime: new Date(2015, 11, 5, 10, 00),
EndTime: new Date(2015, 11, 5, 11, 00)
}]
},
appointmentWindowOpen: "onAppointmentWindowOpen"
});
});
}

```

**JAVASCRIPT**

```

function onAppointmentWindowOpen(args) {
args.cancel = true; // prevents the display of default appointment window
}

```

**Through Programmatically**

You can add/edit the appointments dynamically through the public method [saveAppointment](#). It accepts the JSON Object data (either a new or updated appointment object) as its argument.

**HTML**

```

<button id="btnAdd" value="Add" onclick="addAppointment()">Add</button>
<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
currentDate: new Date(2015, 11, 7),
appointmentSettings: {
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 06:00 AM"),
EndTime: new Date("2015/11/7 07:00 AM")
}], {

```

```

Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 9:00 AM"),
EndTime: new Date("2015/11/7 02:30 PM")
}]
}
});
});
}

```

## JAVASCRIPT

```

function addAppointment() {
//appointment details
var appointment = {
Subject: "Gym",
StartTime: new Date("2015/11/7 03:30 AM"),
EndTime: new Date("2015/11/7 04:30 AM")
};
//create the schedule object
var schObj = $("#schedule").data("ejSchedule");
//pass the JSON object in the public method
schObj.saveAppointment(appointment);
}

```

## Delete Appointments

The appointments can be deleted in either of the following ways,

- Selecting an appointment and clicking the delete icon in the quick appointment window.
- Hovering the mouse over appointments and clicking on Inline delete option which is enabled by default for all the appointments.
- Selecting an appointment and pressing Delete key.
- Through Programmatically.

A pop-up with a confirmation message will get displayed before deleting an appointment, which can be either switched on/off using the API `showDeleteConfirmationDialog`. Also, the confirmation text in that pop-up can be customized as mentioned [here](#).

**For example**, to localize only the delete confirmation message in the delete window -

## HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

## JAVASCRIPT

```

var deleteCustomizationMessage = {
// customize the confirmation message
DeleteConfirmation: "Do you want to delete this Event?",
// customize the delete window title
MouseOverDeleteTitle: "Delete Event",
// customize the recurrence delete window title

```



```

RecurrenceDeleteTitle: "Delete Repeat Event"
};
// Extend only the required changes to the original locale collection
$.extend(ej.Schedule.Locales["en-US"], deleteCustomizationMessage);

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
width: "100%",
height: "525px",
currentDate: new Date(2015, 11, 5),
showDeleteConfirmationDialog: true,
appointmentSettings: {
dataSource: [{
Id: 101,
Subject: "Talk with Nature",
StartTime: new Date(2015, 11, 5, 10, 00),
EndTime: new Date(2015, 11, 5, 11, 00)
}]
}
});
});
}

```

**Note:** All these CRUD operations on appointments (add/edit/delete) can also be done through the default [context menu](#) options **Add Appointment**, **Edit Appointment** and **Delete Appointment** which is available, when context menu settings is enabled within Scheduler.

### Through Programmatically

You can delete the appointments dynamically using the method [deleteAppointment](#), which accepts the Guid of the appointment or complete appointment data as its argument. The Guid is availed as one of the appointment element's attribute.

### Example 1 - Using GUID

The below code example depicts the way to delete the appointments using GUID programmatically by calling the **deleteAppointment** function within the [appointmentClick](#) event, which triggers whenever the user clicks on an appointment.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {

```

```

currentDate: new Date(2015, 11, 7),
appointmentSettings: {
  //Array of JSON data configure in dataSource
  dataSource: [{
    Id: 1,
    Subject: "Music Class",
    StartTime: new Date("2015/11/7 06:00 AM"),
    EndTime: new Date("2015/11/7 07:00 AM")
  }, {
    Id: 2,
    Subject: "School",
    StartTime: new Date("2015/11/7 9:00 AM"),
    EndTime: new Date("2015/11/7 02:30 PM")
  }]
},
appointmentClick: "onAppointmentClick"
});
});
}

```

## JAVASCRIPT

```

function onAppointmentClick(args) {
  var schObj = $("#schedule").data("ejSchedule");
  schObj.deleteAppointment(args.appointment.Guid);
  // $(".e-appointment").eq(0).attr("guid") --> To get the guid attribute
  // value of an element directly.
}

```

### Example 2 - Using Appointment object

The below code example depicts the way to delete the appointments using appointment data programmatically by calling the **deleteAppointment** function within the [appointmentClick](#) event, which triggers whenever the user clicks on an appointment.

## HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

## JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#schedule"), {
      currentDate: new Date(2015, 11, 7),
      appointmentSettings: {
        //Array of JSON data configure in dataSource
        dataSource: [{
          Id: 1,
          Subject: "Music Class",
          StartTime: new Date("2015/11/7 06:00 AM"),
          EndTime: new Date("2015/11/7 07:00 AM")
        }, {

```

```

Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 9:00 AM"),
EndTime: new Date("2015/11/7 02:30 PM")
}]
},
appointmentClick: "onAppointmentClick"
});
});
}

```

## JAVASCRIPT

```

function onAppointmentClick(args) {
var schObj = $("#schedule").data("ejSchedule");
schObj.deleteAppointment(args.appointment);
}

```

## Handling Appointment Actions

It is possible to define some specific actions to take place before the CRUD operation occurs on the Scheduler appointments through the following available client-side events,

- [beforeAppointmentCreate](#)
- [beforeAppointmentChange](#)
- [beforeAppointmentRemove](#)

**beforeAppointmentCreate** – Triggers before saving a new appointment.

**beforeAppointmentChange** – Triggers when an appointment is edited and before it is being updated to the dataSource.

**beforeAppointmentRemove** – Triggers before deleting an existing appointment.

To stop the save, edit and delete actions on the Scheduler appointments, following code example can be used.

## HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

## JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
width: "100%",
height: "525px",
currentDate: new Date(2015, 11, 5),
appointmentSettings: {
dataSource: [{
Id: 101,

```

```

Subject: "Talk with Nature",
StartTime: new Date(2015, 11, 5, 10, 00),
EndTime: new Date(2015, 11, 5, 11, 00)
}],
},
beforeAppointmentCreate: "onAppointmentSave",
beforeAppointmentChange: "onAppointmentEdit",
beforeAppointmentRemove: "onAppointmentDelete"
});
});
}

```

## JAVASCRIPT

```

function onAppointmentSave(args) {
args.cancel = true; // cancels the save action on appointments.
}
function onAppointmentEdit(args) {
args.cancel = true; // cancels the edit action on appointments.
}
function onAppointmentDelete(args) {
args.cancel = true; // cancels the delete action on appointments.
}

```

## Read Only

An interaction with the appointments of the Scheduler can be enabled/disabled through the [readOnly](#) property. When the `readOnly` property is set to `true`, it is not possible to do any actions on the appointments, but you can navigate between the schedule dates, views and can also be able to see the appointment details in the quick window. By default, this property is set to `false`.

## HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

## JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
//set the Schedule as read only,
readOnly: true,
currentDate: new Date(2015, 11, 7),
appointmentSettings: {
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 09:00 AM"),
EndTime: new Date("2015/11/7 10:00 AM")
}, {
Id: 2,

```

```

Subject: "School",
StartTime: new Date("2015/11/7 02:00 PM"),
EndTime: new Date("2015/11/7 06:30 PM")
}]
}
});
});
}

```

**Note:** When the `readOnly` property is set to true – double clicking the cells will open the appointment window filled with appointment details, which can be allowed to view but cannot be edited or saved.

### Drag and Drop

The appointment time can be modified through the drag and drop behavior, by dragging and dropping it to the new location, so that the start time and end time of the appointment gets changed automatically. We can enable/disable the drag and drop functionality through the `allowDragAndDrop` property. By default, it is set to `true`.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

### JAVASCRIPT

```

/// <reference path="../tsfiles/jquery.d.ts" />
/// <reference path="../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
//disable appointment drag and drop,
allowDragAndDrop: false,
currentDate: new Date(2015, 11, 7),
appointmentSettings: {
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 09:00 AM"),
EndTime: new Date("2015/11/7 10:00 AM")
}, {
Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 02:00 PM"),
EndTime: new Date("2015/11/7 06:30 PM")
}]
}
});
});
}

```

### Handling Drag Actions Dynamically

The drag and drop functionality can be handled with the following three events,

- [dragStart](#)
- [drag](#)
- [dragStop](#)

**dragStart** – Triggers when the appointments are started to drag from its source location.

**drag** – Triggers when the appointments are being dragged over.

**dragStop** – Triggers when the appointments are dropped on a destined location.

The following code example shows how to cancel the dragging functionality with the help of one of these events.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="schedule"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#schedule"), {
      width: "100%",
      height: "525px",
      currentDate: new Date(2015, 11, 5),
      appointmentSettings: {
        dataSource: [{
          Id: 101,
          Subject: "Talk with Nature",
          StartTime: new Date(2015, 11, 5, 10, 00),
          EndTime: new Date(2015, 11, 5, 11, 00)
        }]
      },
      dragStop: "onDragStop"
    });
  });
}
```

### JAVASCRIPT

```
function onDragStop(args) {
  args.cancel = true; // cancels the drag action on appointments.
}
```

### External Drag and Drop

It is possible to drag and drop the external items to and fro the Scheduler control. This action is handled through the property [appointmentDragArea](#), which specifies the draggable area name stating whether the appointments can be dragged outside of the control or within it.

The following code example lets you drag and drop the external items from the tree view control to the Scheduler.

**HTML**

```

<!-- Treeview List -->
<div class="col-md-2">
<span class=""><b>Tutorials </b> </span>
<ul id="treeViewDrag">
<li class="expanded">
HTML
<ul>
<li>Introduction</li>
<li>Editors</li>
<li>Styles</li>
<li>Formatting</li>
<li>Tables</li>
</ul>
</li>
</ul>
</div>
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<div id="customWindow" style="display: none">
<form id="custom">
<table width="100%" cellpadding="5">
<tbody>
<tr>
<td>Subject:</td>
<td colspan="2">
<input id="subject" type="text" value="" name="Subject" style="width:
100%" readonly />
</td>
</tr>
<tr>
<td>Description:</td>
<td colspan="2">
<textarea id="customDescription" name="Description" rows="3" cols="50"
style="width: 100%; resize: vertical"></textarea>
</td>
</tr>
<tr>
<td>StartTime:</td>
<td colspan="2">
<input id="StartTime" type="text" value="" name="StartTime" />
</td>
</tr>
<tr>
<td>EndTime:</td>
<td colspan="2">
<input id="EndTime" type="text" value="" name="EndTime" />
</td>
</tr>
<tr>
<td>Resource:</td>
<td colspan="2">
<input id="resource" type="text" value="" name="Resource" style="width:
100%" readonly />
</td>
</tr>

```

```

<tr style="display: none">
<td>ownerId:</td>
<td colspan="2">
<input id="ownerId" type="text" name="ownerId" />
</td>
</tr>
</tbody>
</table>
</form>
<div>
<button type="submit" onclick="cancel()" id="buttonCancel"
style="float:right;margin-right:20px;margin-bottom:10px;">Cancel</button>
<button type="submit" onclick="save()" id="buttonSubmit"
style="float:right;margin-right:20px;margin-bottom:10px;">Save</button>
</div>
</div>

```

## JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var treeview = new ej.TreeView($("#treeViewDrag"), {
allowDragAndDrop: true,
width: 170,
allowDropChild: false,
allowDropSibling: false,
allowDragAndDropAcrossControl: true,
nodeDragStart: "onDragStart",
nodeDropped: "onDropped",
});
var sample = new ej.Schedule($("#schedule"), {
width: "100%",
height: "525px",
cellWidth: "40px",
showCurrentTimeIndicator: false,
orientation: "horizontal",
views: ["Day", "Week", "WorkWeek", "Month"],
currentDate: new Date(2014, 4, 5),
currentView: ej.Schedule.CurrentView.Workweek,
group: {
resources: ["Owners"]
},
resources: [{
field: "ownerId",
title: "Owner",
name: "Owners", allowMultiple: true,
resourceSettings: {
dataSource: [
{ text: "Nancy", id: 1, groupId: 1, color: "#f8a398" },
{ text: "Steven", id: 3, groupId: 2, color: "#56ca85" },
{ text: "Michael", id: 5, groupId: 1, color: "#51a0ed" },
{ text: "Milan", id: 13, groupId: 3, color: "#99ff99" },
{ text: "Paul", id: 15, groupId: 3, color: "#cc99ff" }
],

```



```

text: "text", id: "id", groupId: "groupId", color: "color"
}
}],
appointmentSettings: {
dataSource:
ej.DataManager(window.HorizontalResourcesTutorials).executeLocal(ej.Query
()).take(10)),
id: "Id",
subject: "Subject",
startTime: "StartTime",
endTime: "EndTime",
description: "Description",
allDay: "AllDay",
recurrence: "Recurrence",
recurrenceRule: "RecurrenceRule",
resourceFields: "ownerId"
},
dragStop: "onDragStop"
});
var button1 = new ej.Button($("#buttonSubmit"), {
width: '85px'
});
var button2 = new ej.Button($("#buttonCancel"), {
width: '85px'
});
var DateTimePicker1 = new ej.DateTimePicker($("#StartTime"), {
width: '150px'
});
var DateTimePicker2 = new ej.DateTimePicker($("#EndTime"), {
width: '150px'
});
var dialog = new ej.Dialog($("#customWindow"), {
width: 600,
height: "auto",
position: { X: 200, Y: 100 },
showOnInit: false,
enableModal: true,
title: "Create Appointment",
enableResize: false,
allowKeyboardNavigation: false,
close: "clearFields"
});
});
}

```

## JAVASCRIPT

```

function onDragStart(e) {
if (e.targetElementData.parentId == "") return false;
}
function onDropped(e) {
if ($(e.target).parents(".e-schedule").length != 0) {
var scheduleObj = $("#Schedule1").data("ejSchedule");
var result = scheduleObj.getSlotByElement($(e.target));
// set value to custom appointment window fields
$("#subject").val(e.droppedElementData.text);
}
}

```

```

$("#customDescription").val(e.droppedElementData.text);
$("#StartTime").ejDateTimePicker({ value: new Date(result.startTime) });
$("#EndTime").ejDateTimePicker({ value: new Date(result.endTime) });
$("#resource").val(result.resources.text);
$("#ownerId").val(result.resources.id);
$("#customWindow").ejDialog("open");
}
}
function save() {
var obj = {};
var formElement = $("#customWindow").find("#custom").get(0);
for (var index = 0; index < formElement.length; index++) {
var columnName = formElement[index].name, $element =
$(formElement[index]);
if (columnName !== undefined) {
if (columnName == "Subject") var value = formElement[index].value;
if (columnName == "Description") value = formElement[index].value;
if (columnName == "StartTime") value = new
Date(formElement[index].value);
if (columnName == "EndTime") value = new Date(formElement[index].value);
if (columnName == "ownerId") value = parseInt(formElement[index].value);
if (columnName != "Resource") obj[columnName] = value;
}
}
$("#customWindow").ejDialog("close");
var object = $("#Schedule1").data("ejSchedule");
object.saveAppointment(obj);
}
function cancel() {
$("#customWindow").ejDialog("close");
}
}

```

## Resize

Resizing an appointment is another way to change its start and end time. Mouse hover on the appointments, so that the resizing handlers gets displayed on either sides of the appointment which allows resizing. The resizing functionality can be enabled/disabled by setting the [enableAppointmentResize](#) property. By default it is set to `true`.

## HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

## JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function) () {
var sample = new ej.Schedule($("#schedule"), {
//disable appointment resizes,
enableAppointmentResize: false,
currentDate: new Date(2015, 11, 7),
appointmentSettings: {
//Array of JSON data configure in dataSource

```

```

dataSource: [{
  Id: 1,
  Subject: "Music Class",
  StartTime: new Date("2015/11/7 09:00 AM"),
  EndTime: new Date("2015/11/7 10:00 AM")
}, {
  Id: 2,
  Subject: "School",
  StartTime: new Date("2015/11/7 02:00 PM"),
  EndTime: new Date("2015/11/7 06:30 PM")
}]
}
});
});
}

```

### Handling Resize Actions Dynamically

The appointment resizing functionality can be handled through the following three events,

- [resizeStart](#)
- [resize](#)
- [resizeStop](#)

**resizeStart** – Triggers when the appointments are started resizing from its original time.

**resize** – Triggers when the appointment resizing is in progress.

**resizeStop** – Triggers when the appointment resizing is done.

The following code example shows how to cancel the resizing functionality with the help of one of these events.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#schedule"), {
      width: "100%",
      height: "525px",
      currentDate: new Date(2015, 11, 5),
      appointmentSettings: {
        dataSource: [{
          Id: 101,
          Subject: "Talk with Nature",
          StartTime: new Date(2015, 11, 5, 10, 00),
          EndTime: new Date(2015, 11, 5, 11, 00)
        }]
      },
    },
  ),

```

```
resizeStart: "onResizeStart"
});
});
}
```

## JAVASCRIPT

```
function onResizeStart(args) {
args.cancel = true; // Blocks the resize action on appointments.
}
```

## Categorization

It allows to differentiate the appointments with various categorize options and individual colors. You can also denote the status of the appointments using this categorize option and can specify your own user-defined category collection. It is also possible to select multiple categorize for a single appointment.

### Categorize Settings

The [categorizeSettings](#) holds the below categorize related properties such as,

- [enable](#) - It accepts true or false value, denoting whether to enable/disable the categorize option. Its default value is `false`.
- [allowMultiple](#) - It enables or disables the multiple selection of categories for each appointments in the appointment window as well as in the context menu. Its default value is `false`.
- [dataSource](#) - Binds the categorize dataSource collection. This property should be assigned with the JSON data array collection or instance of [ej.DataManger](#).

We have below 6 default values for Categorize dataSource collection.

## JAVASCRIPT

```
categorizeSettings: {
dataSource: [
{ text: "Blue Category", id: 1, color: "#43b496", fontColor: "#ffffff" },
{ text: "Green Category", id: 2, color: "#7f993e", fontColor: "#ffffff" },
},
{ text: "Orange Category", id: 3, color: "#cc8638", fontColor: "#ffffff" },
},
{ text: "Purple Category", id: 4, color: "#ab54a0", fontColor: "#ffffff" },
},
{ text: "Red Category", id: 5, color: "#dd654e", fontColor: "#ffffff" },
{ text: "Yellow Category", id: 6, color: "#d0af2b", fontColor: "#ffffff" },
}
]
```

The below categorize fields holds the appropriate column names from the dataSource -

| Field name | Description   |
|------------|---|
| id         | It holds the binding name for id field in the categorize dataSource |

|           |  |
|-----------|--|
| text      | It holds the binding name for text field in the categorize dataSource  |
| color     | It holds the binding name for color field in the categorize dataSource.  |
| fontColor | It holds the binding name for fontColor field in the categorize dataSource. This font color applies for the appointment. |

**HTML**

```
<!--Container for ejScheduler widget-->
<div id="schedule"></div>
```

**JAVASCRIPT**

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
currentDate: new Date(2015, 11, 7),
//Configure the categorize settings
categorizeSettings: {
enable: true, //Enable the categorize options
allowMultiple: true, //enable multiple selection options
//data source collection binding
dataSource: [
{ text: "Blue Category", id: 1, color: "#43b496", fontColor: "#ffffff" },
{ text: "Green Category", id: 2, color: "#7f993e", fontColor: "#ffffff" },
},
{ text: "Orange Category", id: 3, color: "#cc8638", fontColor: "#ffffff" },
},
{ text: "Purple Category", id: 4, color: "#ab54a0", fontColor: "#ffffff" },
},
{ text: "Red Category", id: 5, color: "#dd654e", fontColor: "#ffffff" },
{ text: "Yellow Category", id: 6, color: "#d0af2b", fontColor: "#ffffff" },
},
],
text: "text", //text mapper field
id: "id", //id mapper field
color: "color", //categorize color mapper field
fontColor: "fontColor" //categorize appointment font color mapper field
},
appointmentSettings: {
categorize: "categorize",
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 09:00 AM"),
EndTime: new Date("2015/11/7 10:00 AM"),
categorize: "3",
}, {
Id: 2,
```

```

Subject: "School",
StartTime: new Date("2015/11/7 02:00 PM"),
EndTime: new Date("2015/11/7 06:30 PM"),
categorize: "1,2,6" // Multiple categorize id passing
}]
}
});
});
}

```

### Priority

This option prioritize the appointments based on its importance and it can be differentiated with each individual icons/images. By default, there are some specific set of default priority collection and you can also customize it with your own priority collection.

#### Priority Settings

The [prioritySettings](#) holds the below priority related properties such as,

- [enable](#) - It accepts true or false value, denoting whether to enable/disable the priority option. Its default value is **false**.
- [template](#) – Customize the priority icon/images using template options.
- [dataSource](#) – binds the priority dataSource collection. This property should be assigned with the JSON data array collection or instance of [ej.DataManger](#).

We have below 4 default values for priority dataSource collection.

#### JAVASCRIPT

```

prioritySettings: {
  dataSource: [
    { text: "None", value: "none" },
    { text: "High", value: "high" },
    { text: "Medium", value: "medium" },
    { text: "Low", value: "low" }
  ]
}

```

The below priority fields holds the appropriate column names from the dataSource,

| Field name | Description   |
|------------|---|
| text       | It holds the binding name for text field in the priority dataSource   |
| value      | It holds the binding name for value field in the priority dataSource. |

#### HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>
<script>
$(function() {

```

```

$( "#schedule" ).ejSchedule({
  currentDate: new Date(2015, 11, 7),
  //Configure the categorize settings
  prioritySettings: {
    //enable the priority option
    enable: true,
    //Configure the priority data source
    dataSource: [{
      text: "None",
      value: "none"
    }, {
      text: "High",
      value: "high"
    }, {
      text: "Low",
      value: "low"
    }],
    //mapper field for text
    text: "text",
    //mapper field for value
    value: "value"
  },
  appointmentSettings: {
    //set the priority mapper field for appointment data collection
    priority: "priority",
    //Array of JSON data configure in dataSource
    dataSource: [{
      Id: 1,
      Subject: "Music Class",
      StartTime: new Date("2015/11/7 09:00 AM"),
      EndTime: new Date("2015/11/7 10:00 AM"),
      priority: "low", //pass the priority value
    }, {
      Id: 2,
      Subject: "School",
      StartTime: new Date("2015/11/7 02:00 PM"),
      EndTime: new Date("2015/11/7 06:30 PM"),
      priority: "high" //pass the priority value
    }
  ]
});
});
</script>

```

## JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#schedule"), {
      currentDate: new Date(2015, 11, 7),
      //Configure the categorize settings
      prioritySettings: {
        enable: true, //enable the priority option
        //Configure the priority data source

```

```

dataSource: [
{ text: "None", value: "none" },
{ text: "High", value: "high" },
{ text: "Low", value: "low" }
],
text: "text", //mapper field for text
value: "value" //mapper field for value
},
appointmentSettings: {
//set the priority mapper field for appointment data collection
priority: "priority",
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 09:00 AM"),
EndTime: new Date("2015/11/7 10:00 AM"),
priority: "low", //pass the priority value
}, {
Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 02:00 PM"),
EndTime: new Date("2015/11/7 06:30 PM"),
priority: "high" //pass the priority value
}]
}
});
});
}

```

## Search or Filter Appointments

### Appointment Search

The public method [searchAppointments](#) is used to search the appointments in the Scheduler dataSource. It contains the below four arguments such as search string, search field, filter operator and ignore case.

**searchString** - It is used to search the given word/sentence within the appointment data.

**fields** - It is the field with which the search operation takes place. It's an optional argument.

**filterOperator** – It denotes the filter type like `contains`, `greaterthan` or `lessthan`. It's an optional argument.

**ignoreCase** – It is a boolean value to set the search string as case sensitive or not. It's an optional argument.

### HTML

```

<input id="txtSearch" type="text" />
<input id="btnSearch" class="searchApp" type="button" value="Search" />
<div id="grid1"></div>
<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

### JAVASCRIPT



```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#schedule"), {
      currentDate: new Date(2015, 11, 7),
      appointmentSettings: {
//Array of JSON data configure in dataSource
        dataSource: [{
          Id: 1,
          Subject: "Music Class",
          StartTime: new Date("2015/11/7 06:00 AM"),
          EndTime: new Date("2015/11/7 07:00 AM")
        }, {
          Id: 2,
          Subject: "School",
          StartTime: new Date("2015/11/7 9:00 AM"),
          EndTime: new Date("2015/11/7 02:30 PM")
        }
      ]
    });
  });
}

```

## JAVASCRIPT

```

// To bind the click event to the button
$('.searchApp').bind("click", function () {
  var _searchString = $("#txtSearch").val();
  var schObj = $("#Schedule1").data("ejSchedule");
// method to retrieve the appointment based on search string
  var result = schObj.searchAppointments(_searchString);
  showResult(result, _searchString);
});
// method to show the result in a grid
function showResult(list, _searchString) {
  if (!ej.isNullOrUndefined(list) && list.length != 0 && _searchString != "") {
    $("#grid1").show();
    $("#grid1").data("ejGrid") && $("#grid1").ejGrid("destroy");
    $("#grid1").ejGrid({
      allowScrolling: true,
      dataSource: list,
      allowPaging: true
    });
  }
}

```

### Appointment Filters

The appointments can be filtered or shortlisted based on the simple or complex conditions with four available properties such as **field**, **operator**, **value** and **predicate** which is passed to the public method [filterAppointments](#).

**field** - It is the field, with which the search operation takes place. It's an optional argument.

**operator** – It is generally used to specify the [filter](#) type.

**value** – It is the filter keyword based on which the records are filtered.

**predicate** – To add more than one conditional query, need to use **and**, **or** [predicates](#).

### HTML

```
<input id="btnSearch" class="searchApp" type="button" value="Filter" />
<div id="grid1"></div>
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
currentDate: new Date(2015, 11, 7),
appointmentSettings: {
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 06:00 AM"),
EndTime: new Date("2015/11/7 07:00 AM")
}, {
Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 9:00 AM"),
EndTime: new Date("2015/11/7 02:30 PM")
}]
}
});
});
}
```

### JAVASCRIPT

```
// Method to bind the button click event
$('.searchApp').bind("click", function() {
// Add the filter data as like in the below format
var filter = [
{
field: "Subject", // field configure
operator: "contains",
value: "Music",
predicate: "or" // predicate
}, {
field: "Subject", // field configure
operator: "contains",
value: "School",
predicate: "or" // predicate
}
];
```

```

var schObj = $("#Schedule1").data("ejSchedule");
// Method to get the Filtered appointment
var result = schObj.filterAppointments(filter);
showResult(result);
});
// method to show the result in a grid
function showResult(list) {
if (!ej.isNullOrUndefined(list) && list.length != 0) {
$("#grid1").show();
$("#grid1").data("ejGrid") && $("#grid1").ejGrid("destroy");
$("#grid1").ejGrid({
dataSource: list,
allowPaging: true
});
}
}
}

```

### Recurrence Options

There are scenarios where you require the same appointments to be repeatedly created for multiple days on daily, weekly, monthly, and yearly or on every weekday basis.

In appointment data collection, **recurrence** and **recurrenceRule** are dependent fields. While creating or binding the recurrence appointment, the **recurrence** field is set to **true** and **recurrenceRule** contains recurrence pattern in string format.

#### Recurrence Rule

The recurrence appointments are created based on the recurrence rule. The RecurrenceRule is a string value that contains the details of the recurrence appointments like

- repeat type - daily/weekly/monthly/yearly/every weekday
- how many times it needs to be repeated
- the interval duration
- the time period to render the appointment, etc.,

It has the following properties based on which the recurrence appointments are rendered in the Schedule control with its respective time period.

| S.N<br>o | Property | Purpose  |
|----------|----------|--|
| 1        | FREQ     | Maintains the Repeat type value of the appointment.<br><br>(Example: Daily, Weekly, Monthly, Yearly, Every week day)<br><br>Example: FREQ=DAILY;INTERVAL=1   |
| 2        | INTERVAL | Maintains the interval value of the appointments.<br><br>For example, when you create the daily appointment at an interval of 2, the appointments are rendered on the days Monday, Wednesday and Friday. (Creates an appointment on all days by leaving the interval of one day gap) |

|   |            |   |
|---|------------|---|
|   |            | Example: FREQ=DAILY;INTERVAL=2  |
| 3 | COUNT      | <p>It holds the appointment's count value.</p> <p>For example, When the recurrence appointment count value is 10, which means that 10 instances of appointments are created in the recurrence series.</p> <p>Example: FREQ=DAILY;INTERVAL=1;COUNT=10</p>  |
| 4 | UNTIL      | <p>This property is used to store the recurrence end date value.</p> <p>For example, when you set the end date of appointment as 11/30/2015, the UNTIL property holds the end date value denoting when the recurrence actually ends.</p> <p>Example: FREQ=DAILY;INTERVAL=1;UNTIL=11/30/2015</p>   |
| 5 | BYDAY      | <p>It holds the DAY values, representing on which the appointments actually renders.</p> <p>For example, Create the weekly appointment, and select the day(s) from the day options (Monday/Tuesday/Wednesday/Thursday/Friday/Saturday/Sunday). When Monday is selected, the first two letters of the selected day "MO" is saved in the BYDAY property. When multiple days are selected, the values are separated by commas.</p> <p>Example: FREQ=WEEKLY;INTERVAL=1;BYDAY=MO,WE;COUNT=10</p> |
| 6 | BYMONTHDAY | <p>This property is used to store the date value of the Month, while creating the Month recurrence appointment.</p> <p>For example, when you create a Monthly recurrence appointment for every 3rd day of the month, then BYMONTHDAY holds the value 3 and creates the appointment on 3rd day of every month.</p> <p>Example: FREQ=MONTHLY;BYMONTHDAY=3;INTERVAL=1;COUNT=10</p>   |
| 7 | BYMONTH    | <p>This property is used to store the index value of the selected Month while creating the yearly appointments.</p> <p>For example, when you create the yearly appointment on June month, the index value of June month 6 will get stored in the BYMONTH field. The appointment is created on every 6th month of a year.</p> <p>Example: FREQ=YEARLY;BYMONTHDAY=16;BYMONTH=6;INTERVAL=1;COUNT=10</p>  |
| 8 | BYSETPOS   | This property is used to store the index value of the week.   |

|    |               |   |
|----|---------------|---|
|    |               | <p>For example, when you create the monthly appointment in second week of a month, the index value of the second week (2) is stored in BYSETPOS.</p> <p>Example: <code>FREQ=MONTHLY;BYDAY=MO;BYSETPOS=2;UNTIL=8/11/2015</code></p>  |
| 9  | WKST          | <p>This property is used to store the start day value of a week.</p> <p>For example, when you render the workweek the "WKST" value is Monday.</p>   |
| 10 | EXDATE        | <p>EXDATE is used to hold the modified appointment date details (date value) in the recurrence appointment series.</p> <p>For example, when you change the recurrence appointment instance under the date "6/19/2015", then this date is added to the recurrence rule EXDATE field. "EXDATE" is also used to differentiate the edited occurrence of the recurrence series for some internal process while doing the "Edit Series or Delete series" actions.</p> <p>Example:<br/> <code>FREQ=WEEKLY;BYDAY=MO,TU,WE,TH,FR;COUNT=10;EXDATE=6/18/2015,6/20/2015;RECUREREDITID=1651</code></p> |
| 11 | RECUREREDITID | <p>This property contains the Parent Id value of the edited appointment. It is used to track the edited appointment occurrence with its parent recurrence appointment series.</p> <p>For example, when you edit the particular occurrence of the recurrence appointment series, the "RECUREREDITID" is added to that edited appointment depicting its parent Id.</p> <p>Example:<br/> <code>FREQ=WEEKLY;BYDAY=MO,TU,WE,TH,FR;COUNT=10;EXDATE=6/18/2015,6/20/2015;RECUREREDITID=1651</code></p>  |

To know more about other possible combinations of above specified recurrence rule properties, refer [here](#).

### HTML

```
<!--Container for ejScheduler widget-->
<div id="schedule"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
```

```

var sample = new ej.Schedule($("#schedule"), {
  currentDate: new Date(2015, 11, 7),
  appointmentSettings: {
    //Recurrence mapper field
    recurrence: "Recurrence",
    recurrenceRule: "RecurrenceRule",
    //Array of JSON data configure in dataSource
    dataSource: [{
      Id: 1,
      Subject: "Music Class",
      StartTime: new Date("2015/11/7 06:00 AM"),
      EndTime: new Date("2015/11/7 07:00 AM")
    }, {
      Id: 2,
      Subject: "School",
      StartTime: new Date("2015/11/7 9:00 AM"),
      EndTime: new Date("2015/11/7 02:30 PM"),
      Recurrence: true, //enable recurrence options
      RecurrenceRule: "FREQ=DAILY;INTERVAL=1;COUNT=5" //Recurrence rule.
    }]
  }
});

```

#### Recurrence Validation

The default recurrence validation has been included for recurrence appointments similar to the one available in outlook. The validation occurs during the recurrence appointment creation, drag and drop or resizing of the recurrence appointments and also if any single occurrence changes. The validation can be disabled by setting the [enableRecurrenceValidation](#) property to `false`.

#### HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

#### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#schedule"), {
      currentDate: new Date(2015, 11, 7),
      //disable the recurrence validation
      enableRecurrenceValidation: false,
      appointmentSettings: {
        //Recurrence mapper field
        recurrence: "Recurrence",
        recurrenceRule: "RecurrenceRule",
        //Array of JSON data configure in dataSource
        dataSource: [{
          Id: 1,
          Subject: "Music Class",
          StartTime: new Date("2015/11/7 06:00 AM"),

```

```

EndTime: new Date("2015/11/7 07:00 AM")
}, {
Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 9:00 AM"),
EndTime: new Date("2015/11/7 02:30 PM"),
Recurrence: true, //enable recurrence options
RecurrenceRule: "FREQ=DAILY;INTERVAL=1;COUNT=5" //Recurrence rule.
}]
}
});
});
}

```

**Note:** You can parse the **RecurrenceRule** of an appointment from the server-side by making use of a new generic utility class **RecurrenceHelper**. Refer this [KB document](#).

#### *Recurrence Edit and delete options*

The recurring appointments can be edited or deleted in three ways as below:

- Single occurrence
- Following appointments
- Entire series

#### *Single occurrence*

When an option "Only this Appointment" is selected, a single occurrence of the recurrence appointment alone will be edited or deleted.

#### *Following appointments*

When an option "Following Appointments" is selected, all the following events of the recurrence appointment from the current instance will be edited or deleted. The previous instances of the recurrence appointment before this current instances will be retained as it is on the Scheduler.

#### *Entire series*

The entire recurrence series will be edited / deleted, on selecting this option.

#### *Reminder*

Reminder option notifies all the appointments before some specific time. By default, it notifies before 5 minutes. Each and every appointment triggers the [reminder](#) event and can utilize this event for other user actions like mailing particular event to someone or to do any kind of manipulations with the reminder appointments and so on. The `reminderSettings` includes the following 2 properties namely,

- **enable** - To enable the reminder settings of the Schedule control, set the **enable** property as `true` within the [reminderSettings](#) option.
- **alertBefore** - Accepts the integer value to denote the time, before how long the reminder should be notified to the user.

#### **HTML**

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
//Reminder configuration
reminderSettings: {
enable: true, //enable the reminder options
alertBefore: 10 //notify before 10 minutes
},
timeZone: "UTC 00:00",
reminder: "reminderCustom", //Reminder event configure
appointmentSettings: {
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date(new Date().setMinutes(new Date().getMinutes() + 11)),
// new Date("2015/11/7 06:00 AM"),
EndTime: new Date(new Date().setHours(new Date().getHours() + 1)) // new
Date("2015/11/7 07:00 AM")
}, {
Id: 2,
Subject: "School",
StartTime: new Date(new Date().setHours(new Date().getHours() + 2)),
EndTime: new Date(new Date().setHours(new Date().getHours() + 4))
}]
}
});
});
}

```

**JAVASCRIPT**

```

function reminderCustom(args) {
alert("Reminder Appointment");
}

```

**Note:** Whenever the reminder setting is enabled in the Scheduler with some specific value (in minutes) assigned to the **alertBefore** property, the **reminder** event gets triggered before this specified value. It includes the reminder appointment's entire information within its arguments.

**Block Time Intervals**

It allows to block the particular timeslots in Schedule. When specific timeslots are blocked, the appointments that lies in that range can either be made read-only or else can be allowed to interact with the users based on the value assigned to the `isBlockAppointment` property.

**Block Settings**

The [blockoutSettings](#) holds the below block intervals related properties such as,

- [enable](#) - It accepts true or false value, denoting whether to enable/disable the block intervals option. It's default value is `false`.



- [templateId](#) â€” It applies the template design to block the intervals.
- [dataSource](#) â€” Binds the block intervals dataSource collection. This property should be assigned either with the JSON data array collection or instance of [ej.DataManger](#).

The below `blockoutSettings` fields holds the appropriate column names from the dataSource -

| Field name         | Description   |
|--------------------|---|
| id                 | It holds the binding name for id field in the dataSource of <code>blockoutSettings</code>                   |
| subject            | It holds the binding name for subject field in the dataSource of <code>blockoutSettings</code>              |
| startTime          | It holds the binding name for startTime field in the dataSource of <code>blockoutSettings</code> .          |
| endTime            | It holds the binding name for endTime field in the dataSource of <code>blockoutSettings</code> .            |
| isBlockAppointment | It holds the binding name for isBlockAppointment field in the dataSource of <code>blockoutSettings</code> . |
| isAllDay           | It holds the binding name for isAllDay field in the dataSource of <code>blockoutSettings</code> .           |
| resourceId         | It holds the binding name for resourceId field in the dataSource of <code>blockoutSettings</code> .         |
| customStyle        | It holds the binding name for customStyle field in the dataSource of <code>blockoutSettings</code> .        |

### HTML

```
<!--Container for ejScheduler widget-->
<div id="schedule"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#schedule"), {
      currentDate: new Date(2014, 4, 5),
      //Configure the block settings
      blockoutSettings: {
        //Enable the block options
        enable: true,
        //data source collection binding
      }
    });
  });
}
```

```

dataSource: [{
  BlockId: 101,
  BlockStartTime: new Date(2014, 4, 5, 10, 00),
  BlockEndTime: new Date(2014, 4, 5, 11, 00),
  BlockSubject: "Service",
  IsBlockAppointment: true
}, {
  BlockId: 102,
  BlockStartTime: new Date(2014, 4, 4, 12, 00),
  BlockEndTime: new Date(2014, 4, 4, 13, 00),
  BlockSubject: "Maintenance",
  IsBlockAppointment: true
}],
id: "BlockId", //id mapper field
startTime: "BlockStartTime", //start time mapper field
endTime: "BlockEndTime", //end time mapper field
subject: "BlockSubject", //subject mapper field
isBlockAppointment: "IsBlockAppointment" //block appointment mapper field
}
});
});
}

```

### Blocking Appointments

The Appointments that lies within the blocked time range can be restricted to perform CRUD operations in it and can be made read-only. This can be achieved by setting [isBlockAppointment](#) property to true.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#schedule"), {
      currentDate: new Date(2014, 4, 5),
      //Configure the block settings
      blockoutSettings: {
        //Enable the block options
        enable: true,
        //data source collection binding
        dataSource: [{
          BlockId: 101,
          BlockStartTime: new Date(2014, 4, 5, 10, 00),
          BlockEndTime: new Date(2014, 4, 5, 11, 00),
          BlockSubject: "Service",
          IsBlockAppointment: true
        }],
        id: "BlockId",
        startTime: "BlockStartTime",
        endTime: "BlockEndTime",

```

```

subject: "BlockSubject",
//Bind the block appointment field in datasource
isBlockAppointment: "IsBlockAppointment"
}
});
});
}

```

### Customizing block time intervals

The [blockoutSettings](#) holds the below properties to customize the block intervals such as,

- [templateId](#) - Template design that applies on the block intervals.
- [customStyle](#) - The custom CSS that applies on the blocked intervals.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="schedule"></div>
<!--Template to apply block intervals-->
<script id="blockTemplate" type="text/x-jsrender">
<div style="height:100%">
<div>{:BlockSubject}</div>
</div>
</script>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
currentDate: new Date(2014, 4, 5),
blockoutSettings: {
enable: true,
templateId: "#blockTemplate",
dataSource: [{
BlockId: 101,
BlockStartTime: new Date(2014, 4, 5, 10, 00),
BlockEndTime: new Date(2014, 4, 5, 11, 00),
BlockSubject: "Service",
IsBlockAppointment: true
}],
id: "BlockId",
startTime: "BlockStartTime",
endTime: "BlockEndTime",
subject: "BlockSubject",
isBlockAppointment: "IsBlockAppointment"
}
});
});
}

```

*Blocking time interval based on resources*

- [resourceId](#) - property used within the `blockoutSettings` which accepts the resource id's can be used to apply the block intervals based on the resources.

**HTML**

```
<!--Container for ejScheduler widget-->
<div id="schedule"></div>
```

**JAVASCRIPT**

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#schedule"), {
currentDate: new Date(2014, 4, 2),
group: {
resources: ["Owners"]
},
resources: [{
field: "ownerId", title: "Owner", name: "Owners",
allowMultiple: true,
resourceSettings: {
dataSource: [
{ text: "Nancy", id: 1, color: "#f8a398" },
{ text: "Steven", id: 2, color: "#56ca85" }
],
text: "text", id: "id", color: "color"
}
}],
appointmentSettings: {
dataSource: [{
EventId: 100,
EventSubject: "Research on Sky Miracles",
EventStartTime: new Date(2014, 4, 2, 9, 00),
EventEndTime: new Date(2014, 4, 2, 10, 30),
ownerId: 2
}],
id: "EventId",
startTime: "EventStartTime",
endTime: "EventEndTime",
subject: "EventSubject",
resourceFields: "ownerId"
},
//Configure the block settings
blockoutSettings: {
//Enable the block options
enable: true,
//data source collection binding
dataSource: [{
BlockId: 101,
BlockStartTime: new Date(2014, 4, 1, 10, 00),
BlockEndTime: new Date(2014, 4, 1, 11, 00),
```

```

BlockSubject: "Travel",
IsBlockAppointment: true,
BlockResId: 2
}],
id: "BlockId",
startTime: "BlockStartTime",
endTime: "BlockEndTime",
subject: "BlockSubject",
isBlockAppointment: "IsBlockAppointment",
//resource id mapper field
resourceId: "BlockResId"
}
});
});
}

```

## Context Menu

Scheduler provides default context menu options for both appointments as well as work cells. It also allows to define additional custom context menu options.

The options that are available under [contextMenuSettings](#) are as follows,

- **enable** - Enables/disables the context menu option in Scheduler.
- **menuItems** - Contains the sub-menu collections related to both the appointment and cells.

## Default Menu Options

The menu items contains two separate collection based on the appointment and cells.

### Appointment

The appointment collection includes the following options.

|                              |
|------------------------------|
| Open Appointment (default)   |
| Delete Appointment (default) |
| Print Appointment            |
| Categorize                   |

### Cells

The default options available within the cell collection includes -

|                                       |
|---------------------------------------|
| New Appointment                       |
| New Recurring Appointment             |
| Today                                 |
| Go to date                            |
| Settings (View, TimeMode, Work Hours) |

The following code snippet shows how to enable the context menu settings in Scheduler and to make use of it with default available options.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
contextMenuSettings: {
enable: true,
menuItems: {
appointment: [
{ id: "open", text: "Open Appointment" },
{ id: "delete", text: "Delete Appointment" }
],
cells: [
{ id: "new", text: "New Appointment" },
{ id: "recurrence", text: "New Recurring Appointment" },
{ id: "today", text: "Today" },
{ id: "goToDate", text: "Go to date" },
{ id: "settings", text: "Settings" },
{ id: "view", text: "View", parentId: "settings" },
{ id: "timeMode", text: "TimeMode", parentId: "settings" },
{ id: "view_Day", text: "Day", parentId: "view" },
{ id: "view_Week", text: "Week", parentId: "view" },
{ id: "view_Workweek", text: "Workweek", parentId: "view" },
{ id: "view_Month", text: "Month", parentId: "view" },
{ id: "timeMode_Hour12", text: "12 Hours", parentId: "timeMode" },
{ id: "timeMode_Hour24", text: "24 Hours", parentId: "timeMode" },
{ id: "workhours", text: "Work Hours", parentId: "settings" }
]
}
},
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30)
}]
}
});
});
}
```

---

**Note:** In agenda view, only the appointment menu items shows up in the context menu options. For default menu items, the id must be defined the same as mentioned in the above code example – as we have processed the menus based on this id within our source.

---

### Custom Menu Options

Apart from the default available options, it is also possible to add custom menu options to the context-menu of both the appointment and cell collection.

The following code example depicts how **to add the custom menu items** to the appointment and cell collection of the context menu settings.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
contextMenuSettings: {
enable: true,
menuItems: {
appointment: [
{ id: "open", text: "Open Appointment" },
{ id: "delete", text: "Delete Appointment" },
{ id: "option1", text: "User Option 1" }
],
cells: [{
id: "celloption1",
text: "Custom Option 1"
}]
}
},
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30)
}]
}
});
});
}
```

---

**Note:** The **id** given for the custom menu items **must be unique** in both the appointment and cell collection.

---

## Handling Menu Actions

To define specific actions for a click made on the custom menu items, the client-side event [menuItemClick](#) can be used as depicted in the below code example.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
contextMenuSettings: {
enable: true,
menuItems: {
appointment: [
{ id: "open", text: "Open Appointment" },
{ id: "delete", text: "Delete Appointment" },
{ id: "option1", text: "User Option 1" }
]
}
},
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30)
}]
},
menuItemClick: "onMenuItemClick"
});
});
}
```

### JAVASCRIPT

```
function onMenuItemClick(args) {
//args.events contains information of the clicked menu item.
if (args.events.ID == "option1")
alert("Custom menu clicked");
}
```

Also, it is possible to predict the target on which the right click is made, either on the cells or appointments with the use of the event [beforeContextMenuOpen](#). The below code example shows how to block the display of context menu, when right clicked on the cells by setting **args.cancel** as **true**.

### HTML

```
<!--Container for ejScheduler widget-->
```



```
<div id="Schedule1"></div>
```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
contextMenuSettings: {
enable: true,
menuItems: {
appointment: [{
id: "open",
text: "Open Appointment"
}, {
id: "delete",
text: "Delete Appointment"
}, {
id: "option1",
text: "User Option 1"
}]
}
},
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30)
}]
}
beforeContextMenuOpen: "onBeforeContextMenuOpen"
});
});
}

```

### JAVASCRIPT

```

function onBeforeContextMenuOpen(args) {
//args.events.target - target information to depict either
cell/appointment
if ($(args.events.target).hasClass("e-workcells") ||
$(args.events.target).hasClass("e-monthcells"))
args.cancel = true;
}

```

### Adding Categorize Option

To include the default categorize option within the context menu, it is necessary to enable the [categorizeSettings](#) property as shown in the below code example.

### HTML

```
<!--Container for ejScheduler widget-->
```

```
<div id="Schedule1"></div>
```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
contextMenuSettings: {
enable: true,
menuItems: {
appointment: [{
id: "open",
text: "Open Appointment"
}, {
id: "delete",
text: "Delete Appointment"
}, {
id: "categorize",
text: "Categorize"
}]
}
},
categorizeSettings: {
enable: true
},
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30)
}]
}
});
});
}

```

**Note:** The **categorize** option must be added only to the **appointment** collection, which displays on right clicking the appointments.

### Remote Data Binding for Categorize

In Schedule, we can also binding the categorize datasource using remote data. The below code example shows how to bind the datasource to categorizeSettings.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
```

```

/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var dataManager = ej.DataManager({
url: "Home/GetCategorizeData",
adaptor: new ej.UrlAdaptor()
});
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
categorizeSettings: {
enable: true,
allowMultiple: true,
dataSource: dataManager,
id: "Id",
fontColor: "FontColor",
color: "Color",
text: "Text"
},
appointmentSettings: {
categorize: "Categorize",
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Categorize: "1"
}]
}
});
});
}

```

The server-side controller code to handle the CRUD operations are as follows.

### C#

```

public JsonResult GetCategorizeData()
{
// ScheduleDataDataContext is a LINQ-to-SQL data class name that is
defined in the .dbml file to access the tables from the database
IEnumerable data = new ScheduleDataDataContext().CategoryData;
return Json(data, JsonRequestBehavior.AllowGet);
}

```

## Resources

The Scheduler provides **Resources** support, with which the single Scheduler is shared by multiple resources simultaneously. Each resource in the Scheduler is arranged in a column/row wise, with individual spacing to display all its respective appointments on a single page. It also supports the grouping of resources, thus enabling the categorization of resources in a hierarchical structure and shows it either in expandable groups (**Horizontal view**) or else vertical hierarchy one after the other (**Vertical view**).

One or more resources can be assigned to the Scheduler appointments by making selection of the resource options available in the appointment window.

## Fields of Resources

The default options available within the [resources](#) collection are as follows,

### *name (\*\*String\*\*)*

A unique resource name which is used for differentiating various resource objects while grouping it in levels.

### *title (\*\*String\*\*)*

It holds the title name of the resource field to be displayed on the Scheduler appointment window.

### *field (\*\*String\*\*)*

It holds the name of the resource field to be bound to the Scheduler appointments which contains the resource Id.

### *allowMultiple (\*\*Boolean\*\*)*

When set to true, allows multiple selection of resource names, thus creating multiple instances of same appointment for the selected resources.

### *resourceSettings (\*\*Object\*\*)*

It holds the field names of the resources dataSource to be bound to the Scheduler.

The following are the resource fields which must be defined within the **resourceSettings** that holds the appropriate column names from the dataSource.

| Field name       | Description   |
|------------------|---|
| text             | Binds the text field name in the dataSource to the resourceSettings text. These text gets listed out in the resources field of the appointment window. Itâ€™s mandatory.                                  |
| id               | Binds the id field name in the dataSource to the resourceSettings id. Itâ€™s mandatory.   |
| groupId          | Binds the groupId field name in the dataSource to the resourceSettings groupId. This field is not necessary for a resource object (resource data) defined as first level within the resources collection. |
| color            | Binds the color field name in the dataSource to the resourceSettings color. It is optional.   |
| appointmentClass | Binds the appointmentClass field name in the dataSource. It applies the custom CSS class name to the appointments based on the resources.   |
| start            | Binds the starting work hour field name in the dataSource. It's optional, but when provided with some numeric value will set the starting work hour for specific resources.                               |
| end              | Binds the end work hour field name in the dataSource. It's optional, but when provided with some numeric value will set the end work hour for specific resources.   |

|          |  |
|----------|--|
| workWeek | Binds the resources working days field name in the dataSource. It's optional, and accept array of strings which is nothing but only the week day names. When provided with some values (array of day names), only those days will render for the specific resources. |
|----------|--|

**Example:** To set the resources options using all the above specified fields,

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
currentDate: new Date(2015, 04, 05),
resources: [{
field: "ownerId",
title: "Owner",
resourceSettings: {
dataSource: [
{ OwnerText: "Nancy", id: 1, OwnerColor: "#f8a398" },
{ OwnerText: "Steven", id: 2, OwnerColor: "#56ca95" }
],
text: "OwnerText",
id: "id",
color: "OwnerColor"
}
}, {
field: "roomId",
title: "Room(s)",
resourceSettings: {
dataSource: [
{ text: "Room1", id: 1, groupId: 1, color: "#f8a398", workHourStart: 10,
workHourEnd: 18, customDays: ["monday", "wednesday", "friday"] },
{ text: "Room2", id: 2, groupId: 2, color: "#56ca85", workHourStart: 6,
workHourEnd: 10, customDays: ["monday", "saturday"] },
{ text: "Room3", id: 3, groupId: 2, color: "#56ac88", workHourStart: 11,
workHourEnd: 15, customDays: ["tuesday", "friday"] }
],
text: "text",
id: "id",
color: "color",
groupId: "groupId",
start: "workHourStart",
end: "workHourEnd",
workWeek: "customDays"
}
```

```

    }
  }],
  appointmentSettings: {
    dataSource: [{
      Id: 100,
      Subject: "Research on Sky Miracles",
      StartTime: new Date(2015, 04, 05, 9, 00),
      EndTime: new Date(2015, 04, 05, 10, 30),
      ownerId: 2,
      roomId: 3
    }],
    resourceFields: "ownerId,roomId"
  }
});
});
}

```

**Note:** The resource object defined at **first level** within the **resources** collection doesn't make use of the **groupId** field, as there is no previous levels applicable to map.

### Data Binding

The resource data can be bound to the Schedule control through the **resourceSettings** options available within the **resources** property. The data-binding can be done either using JSON object collection or DataManager ([ej.DataManager](#)) instance which contains the resources related data.

### JSON Data

**Example:** To set the resource data with array of **JSON** object collection

### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#Schedule1"), {
      width: "100%",
      currentDate: new Date(2015, 04, 05),
      resources: [{
        field: "ownerId",
        title: "Owner",
        resourceSettings: {
          dataSource: [{
            text: "Nancy",
            id: 1,
            color: "#f8a398"
          }, {
            text: "Steven",
            id: 2,
            color: "#56ca85"
          }],

```

```

text: "text",
id: "id",
color: "color"
}
}],
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 04, 05, 9, 00),
EndTime: new Date(2015, 04, 05, 10, 30),
ownerId: 2
}, {
Id: 101,
Subject: "Research on Clouds",
StartTime: new Date(2015, 04, 07, 7, 00),
EndTime: new Date(2015, 04, 07, 10, 30),
ownerId: 1
}],
resourceFields: "ownerId"
}
});
});
}

```

#### Remote Data

**Example:** To set the resource data through the instance of `ejDataManager`

#### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

#### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Scheduler($("#Schedule1"), {
width: "60%",
height: "550px",
currentDate: new Date(2015, 04, 05),
resources: [{
field: "ownerId",
title: "Owner",
resourceSettings: {
// referring data from remote service (url binding)
dataSource: ej.DataManager({ url:
"http://mvc.syncfusion.com/OdataServices/Northwnd.svc" }),
// query to fetch the records from the specified table "Events"
query: ej.Query().select("CategoryID",
"CategoryName").from("Categories").take(3),
text: "CategoryName",
id: "CategoryID"
}
}
}
}

```

```

    }],
    appointmentSettings: {
        dataSource: [{
            Id: 100,
            Subject: "Research on Sky Miracles",
            StartTime: new Date(2015, 04, 05, 9, 00),
            EndTime: new Date(2015, 04, 05, 10, 30),
            ownerId: 2
        }, {
            Id: 101,
            Subject: "Research on Clouds",
            StartTime: new Date(2015, 04, 07, 7, 00),
            EndTime: new Date(2015, 04, 07, 10, 30),
            ownerId: 1
        }],
        resourceFields: "ownerId"
    }
});
});
}

```

### Multiple Resources (Without Grouping)

It is possible to display the Scheduler in default look without visually showcasing all the resources on it, but it allow the user to assign the required resources to the appointments through the appointment window resource options.

The appointments belonging to all the resources will be displayed on the Scheduler which will be differentiated based on the resource color assigned in the **resourceSettings** (depicting to which resource that particular appointment belongs).

**Example:** To display default Scheduler with multiple resource options in the appointment window,

### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
    $(function () {
        var sample = new ej.Schedule($("#Schedule1"), {
            width: "100%",
            currentDate: new Date(2015, 04, 05),
            resources: [{
                field: "ownerId",
                title: "Owner",
                allowMultiple: true,
                resourceSettings: {
                    dataSource: [{
                        text: "Nancy",
                        id: 1,
                        color: "#f8a398"
                    }
                ]
            }
        ]
    });
    });
}

```



```

}, {
  text: "Steven",
  id: 2,
  color: "#56ca85"
}],
text: "text",
id: "id",
color: "color"
}
}],
appointmentSettings: {
  dataSource: [{
    Id: 100,
    Subject: "Research on Sky Miracles",
    StartTime: new Date(2015, 04, 05, 9, 00),
    EndTime: new Date(2015, 04, 05, 10, 30),
    ownerId: 2
  }, {
    Id: 101,
    Subject: "Research on Clouds",
    StartTime: new Date(2015, 04, 07, 6, 00),
    EndTime: new Date(2015, 04, 07, 9, 30),
    ownerId: 1
  }],
  resourceFields: "ownerId"
}
});
});
}

```

**Note:** Setting **allowMultiple** to **true** in the above code snippet allows the user to select multiple resources in the appointment window and also creates multiple copies of the same appointment in the Scheduler for each resources while saving.

### Grouping

Scheduler supports both single and multiple levels of resource grouping that can be customized either in horizontal or vertical Scheduler views. In Vertical view - the levels are displayed in a tree structure one after the other, but in horizontal view – the levels are grouped in a vertically expandable/collapsible structure.

#### Single-Level

This type of grouping allows the Scheduler to display all the resources at a single level simultaneously. The appointments will make use of the **color** defined for the first resource instance as its background color.

**Example:** To display the Scheduler with single level resource grouping options,

#### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

#### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />

```

```

/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
currentDate: new Date(2015, 04, 05),
group: {
resources: ["Owners"]
},
resources: [{
field: "ownerId",
title: "Owner",
name: "Owners",
resourceSettings: {
dataSource: [{
text: "Nancy",
id: 1,
color: "#f8a398"
}, {
text: "Steven",
id: 2,
color: "#56ca85"
}],
text: "text",
id: "id",
color: "color"
}
}],
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 04, 05, 9, 00),
EndTime: new Date(2015, 04, 05, 10, 30),
ownerId: 2
}, {
Id: 101,
Subject: "Discovery of Exoplanets",
StartTime: new Date(2015, 04, 07, 6, 00),
EndTime: new Date(2015, 04, 07, 9, 30),
ownerId: 1
}],
resourceFields: "ownerId"
}
});
});
}

```

**Note:** The **name** field mentioned in the **resource** object needs to be specified within the **group** property in order to enable the grouping option in Scheduler.

#### Multi-Level

This type of grouping displays the resources in the Scheduler at multiple levels with a set of resources grouped under each parent. The appointments will make use of the **color** defined for the first/top level resource instance as its background color.

**Example:** To display the Scheduler with multiple level resource grouping options,

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function) () {
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
currentDate: new Date(2015, 04, 05),
group: {
resources: ["Owners", "Rooms"]
},
resources: [{
field: "ownerId",
title: "Owner",
name: "Owners",
resourceSettings: {
dataSource: [{
OwnerText: "Nancy",
id: 1,
OwnerColor: "#f8a398"
}, {
OwnerText: "Steven",
id: 2,
OwnerColor: "#56ca95"
}],
text: "OwnerText",
id: "id",
color: "OwnerColor"
}
}, {
field: "roomId",
title: "Room(s)",
name: "Rooms",
resourceSettings: {
dataSource: [{
text: "Room1",
id: 1,
groupId: 1,
color: "#f8a398"
}, {
text: "Room2",
id: 2,
groupId: 2,
color: "#56ca85"
}, {
text: "Room3",
id: 3,
groupId: 2,
color: "#56ac88"
}
}
}
}
}
```

```

    }],
    text: "text",
    id: "id",
    color: "color",
    groupId: "groupId"
  }
}],
appointmentSettings: {
  dataSource: [{
    Id: 100,
    Subject: "Research on Sky Miracles",
    StartTime: new Date(2015, 04, 05, 9, 00),
    EndTime: new Date(2015, 04, 05, 10, 30),
    ownerId: 2,
    roomId: 3
  }],
  resourceFields: "ownerId,roomId"
}
});
});
}

```

**Note:** Here, the appointments will make use of the **color** defined for the Owners resource instance as its background color.

### Different Working days and Hours for Resources

It is possible to assign different workdays and workhours for each resources present within the Scheduler. The process of assigning different working days for every individual resources is applicable only for the vertical Scheduler mode and not for timeline view, whereas the customization of workhours for each resources is applicable on both the Scheduler orientation. The custom workdays and workhours needs to be defined within the **resourceSettings** property using the following 3 sub-properties available within it.

- [start](#) is used to define the work start hour for each individual resources.
- [end](#) is used to define the work end hour for each individual resources.
- [workWeek](#) is used to define different working days for each individual resources.

**Example:** To display the Scheduler with each individual resources having different workhours and workdays, the code example is depicted below.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#Schedule1"), {
      width: "100%",

```

```

height: "525px",
currentDate: new Date(2014, 4, 5),
currentView: ej.Schedule.CurrentView.Workweek,
group: {
resources: ["Rooms", "Owners"]
},
resources: [{
field: "roomId",
title: "Room",
name: "Rooms", allowMultiple: false,
resourceSettings: {
dataSource: [
{ text: "ROOM1", id: 1, groupId: 1, color: "#cb6bb2" },
{ text: "ROOM2", id: 2, groupId: 1, color: "#56ca85" }
],
text: "text", id: "id", groupId: "groupId", color: "color"
}
}, {
field: "ownerId",
title: "Owner",
name: "Owners", allowMultiple: true,
resourceSettings: {
dataSource: [
{ text: "Nancy", id: 1, groupId: 1, color: "#ffaa00", on: 10, off: 18,
customDays: ["monday", "wednesday", "friday"] },
{ text: "Steven", id: 3, groupId: 2, color: "#f8a398", on: 6, off: 10,
customDays: ["tuesday", "thursday"] },
{ text: "Michael", id: 5, groupId: 1, color: "#7499e1", on: 11, off: 15,
customDays: ["sunday", "tuesday", "thursday", "saturday"] }
],
text: "text", id: "id", groupId: "groupId", color: "color", start: "on",
end: "off", workWeek: "customDays"
}
}],
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 04, 05, 9, 00),
EndTime: new Date(2015, 04, 05, 10, 30),
ownerId: 2,
roomId: 3
}],
resourceFields: "ownerId,roomId"
}
});
});
}

```

## Customization

The Scheduler can be customized in various aspects like -

- Setting different Start/end hour limits
- Highlighting the working hours
- Setting different [date format](#)

- Specifying minimum and maximum date ranges
- Customize the entire appointment window with the user required fields
- Setting different time Slot duration
- Complete Scheduler customization using queryCellInfo event
- Setting different [first day of week](#)

### Hour Customization

It includes customization of displaying Scheduler with specific Start/End hours and also defining the working hour time range which is differentiated as business hours.

### Schedule Display Hours

It denotes the start and end hour time limits to be displayed on the Scheduler. To set this time limit, two properties namely [startHour](#) and [endHour](#) can be used.

- **startHour** - The hour from which the Scheduler time display actually starts.
- **endHour** - The hour on which the Scheduler time display should end.

The following code example renders the scheduler from 7.00 AM to 6.00 PM.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
    $(function () {
        var sample = new ej.Schedule($("#Schedule1"), {
            width: "100%",
            currentDate: new Date(2015, 04, 05),
            startHour: 7,
            endHour: 18,
            appointmentSettings: {
                dataSource: [{
                    Id: 101,
                    Subject: "Talk with Nature",
                    StartTime: new Date(2015, 11, 5, 10, 00),
                    EndTime: new Date(2015, 11, 5, 11, 00)
                }]
            }
        });
    });
}
```

### Working Hours

Working hours indicates the work hour limit within the Scheduler, which is highlighted visually with white colored work cells. To enable the highlighting of work hours on the Scheduler, set the **highlight** option available within the workHours property to **true**. By default, it is set to true. [workHour](#) is a object property which contains the below specified options,

- [highlight](#) – enables/disables the highlighting of work hours.
- [start](#) - sets the start time of the working/business hour in a day.
- [end](#) - sets the end time limit of the working/business hour in a day.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
workHours: {
highlight: true,
start: 8,
end: 16
},
currentDate: new Date(2015, 11, 5),
appointmentSettings: {
dataSource: [{
Id: 101,
Subject: "Talk with Nature",
StartTime: new Date(2015, 11, 5, 10, 00),
EndTime: new Date(2015, 11, 5, 11, 00)
}]
}
});
});
}
```

**Note:** By default, work hour **start** is set to **9** and **end** is set to **18**. Also, the Scheduler cells automatically scrolls up or down based on the starting work hour, to make the user to view that particular time initially.

### TimeScale

The [TimeScale](#) allows the user to set the required time slot duration for the work cells that displays on the Scheduler. It provides option to customize both the major and minor slots using template option. It includes the below properties such as,

- [enable](#) - It accepts true or false value, denoting whether to show or hide the time slots. Its default value is `true`.
- [majorSlot](#) – Specifies the major time slot duration.
- [minorSlotCount](#) – Specifies the value, based on which the minor time slots are divided into appropriate count.
- [TimeScale templates](#) - 2 template options available for customizing timeScales namely `minorSlotTemplateId` and `majorSlotTemplateId`.

The majorSlot and minorSlot can be set on the Scheduler with the following code example.

## HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

## JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
timeScale: {
enable: true,
majorSlot: 60,
minorSlotCount: 6
},
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});
}
```

### Hide Weekend days

The Scheduler can be customized to display only the working days, thus hiding the weekend days from it. The working days render based on the values given in the [workWeek](#) property. The days that are not mentioned in the [workWeek](#) collection is considered to be the weekend days and it can be hidden from the Scheduler by setting `false` to the [showWeekend](#) property.

The following code example renders the Scheduler by hiding the weekend days.

## HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

## JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
currentDate: new Date(2015, 04, 05),
showWeekend: false,
appointmentSettings: {
```



```

dataSource: [{
  Id: 101,
  Subject: "Talk with Nature",
  StartTime: new Date(2015, 11, 5, 10, 00),
  EndTime: new Date(2015, 11, 5, 11, 00)
}]
}
});
});
}

```

### Date Customization

The date in the Scheduler can be customized by setting specific minimum and maximum date ranges and also defining various date formats to it.

#### Current Date

The Current date indicates the date with which the Scheduler loads initially and based on which the appropriate date range displays in the week/workweek/month/agenda views. To set the current date to the Scheduler – use the following code example,

#### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

#### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#Schedule1"), {
      currentDate: new Date(2015, 11, 5),
      appointmentSettings: {
        dataSource: [{
          Id: 101,
          Subject: "Talk with Nature",
          StartTime: new Date(2015, 11, 5, 10, 00),
          EndTime: new Date(2015, 11, 5, 11, 00)
        }]
      }
    });
  });
}

```

---

**Note:** By default, the System current date will be taken as Scheduler's current date.

---

#### MinDate and MaxDate

Providing the [minDate](#) and [maxDate](#) property with some date values, allows the Scheduler to set the minimum and maximum date range. The Scheduler date that lies beyond these minimum and maximum date range will be in a disabled state, so that the date navigation is blocked beyond these specified date range. Also, the appointments that lies beyond these date ranges will not be displayed on the Scheduler.

The following code example shows how to set the minDate and maxDate properties of the Scheduler.

**HTML**

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

**JAVASCRIPT**

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
minDate: new Date(2015, 11, 4),
maxDate: new Date(2015, 11, 8),
appointmentSettings: {
dataSource: [{
Id: 101,
Subject: "Talk with Nature",
StartTime: new Date(2015, 11, 5, 10, 00),
EndTime: new Date(2015, 11, 5, 11, 00)
}]
}
});
});
}
```

**Note:** The **maxDate** value provided should always be greater than that of **minDate** value.

**Appointment Window Customization**

It is possible to use the custom appointment window option to design it with the user-required extra fields apart from the other default available fields. To make use of the customized appointment window, it is necessary to use the [appointmentWindowOpen](#) event within which the display of default appointment window is prevented.

The following code example lets you create the custom appointment window (using recurrence editor feature) with a single extra field for defining the appointment type.

**HTML**

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<div id="customWindow" style="display: none">
<div id="appWindow">
<form id="custom">
<table width="100%" cellpadding="5">
<tbody>
<tr style="display: none">
<td>
Id:
</td>
<td colspan="2">
<input id="customId" type="text" name="Id" />
</td>
</tr>
<tr>
<td>
```

```

Subject:
</td>
<td colspan="2">
<input id="subject" type="text" value="" name="Subject" onfocus="temp()"
style="width: 100%" />
</td>
</tr>
<tr>
<td>
Description:
</td>
<td colspan="2">
<textarea id="customdescription" name="Description" rows="3" cols="50"
style="width: 100%; resize: vertical"></textarea>
</td>
</tr>
<tr>
<td>
StartTime:
</td>
<td>
<input id="StartTime" type="text" value="" name="StartTime" />
</td>
</tr>
<tr>
<td>
EndTime:
</td>
<td>
<input id="EndTime" type="text" value="" name="EndTime" />
</td>
</tr>
<tr>
<td>Appointment Type:</td>
<td><input type="text" id="AppointmentType" /></td>
</tr>
<tr>
<td colspan="3">
<div class="customcheck">AllDay:</div>
<div class="customcheck">
<input id="allDay" type="checkbox" name="AllDay" onchange="allDayCheck()"
/>
</div>
<div class="customcheck">Recurrence:</div>
<div>
<input id="recurrence" type="checkbox" name="Recurrence"
onchange="recurCheck()" />
</div>
</td>
</tr>
<tr id="summaryTr" style="display:none;">
<td colspan="3">
<div class="recSummary">Summary:</div>
<div>
<label id="recSummary" name="Summary"></label>
</div>
</td>

```

```

</tr>
<tr id="editor" style="display:none;">
<td colspan="3">
<div><a id="recredit" onclick="recurrenceRule()">Edit</a></div>
</td>
</tr>
</tbody>
</table>
</form>
<div>
<button type="submit" onclick="cancel()" id="buttonCancel"
style="float:right;margin-right:20px;margin-bottom:10px;">Cancel</button>
<button type="submit" onclick="save()" id="buttonSubmit"
style="float:right;margin-right:20px;margin-bottom:10px;">Submit</button>
</div>
</div>
<div id="recWindow" style="display: none">
<div id="recurrenceEditor"></div>
<br />
<div>
<button type="submit" id="recCancel">Cancel</button>
<button type="submit" id="recSubmit">Submit</button>
</div>
</div>
</div>

```

The styles to be applied for the controls within the custom appointment window are as follows.

#### HTML

```

<style>
.customcheck {
float: left;
margin-right: 10px;
}
.error {
background-color: #FF8A8A;
}
#custom table td {
padding: 5px;
}
</style>

```

Now, define the Schedule control with custom appointment window within script as shown below.

#### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
// defining sub-controls used within custom appointment window
var button1 = new ej.Button($("#buttonSubmit"), {
width: "85px"
});
var button2 = new ej.Button($("#buttonCancel"), {

```

```

width: "85px"
});
var checkbox = new ej.CheckBox($("#recurrence"), {
change: "recurCheck"
});
var datetime1 = new ej.DateTimePicker($("#StartTime"), {
width: "150px"
});
var datetime2 = new ej.DateTimePicker($("#EndTime"), {
width: "150px"
});
var button3 = new ej.Button($("#recSubmit"), {
click: "onRecurrenceClick"
});
var button4 = new ej.Button($("#recCancel"), {
click: "onRecurrenceClick"
});
var dManager = window.Startend;
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
height: "525px",
currentDate: new Date(2014, 4, 5),
appointmentSettings: {
dataSource: dManager,
id: "Id",
startTime: "StartTime",
endTime: "EndTime",
recurrence: "Recurrence",
recurrenceRule: "RecurrenceRule"
},
appointmentWindowOpen: "onAppointmentWindowOpen"
});
// defining recurrence editor control to be used as custom appointment window
var recurrencesample = new ej.RecurrenceEditor($("#recurrenceEditor"), {
selectedRecurrenceType: 0,
frequencies: ["daily", "weekly", "monthly", "yearly", "everyweekday"]
});
// defining dropdown controls to be used within custom appointment window
var dropdownlist = new ej.DropDownList($("#AppointmentType"), {
dataSource: [
{ text: "Tentative", id: 1 },
{ text: "Busy", id: 3 },
{ text: "Free", id: 5 },
{ text: "Out Of Office", id: 7 }
],
fields: {
text: "text",
id: "id",
value: "text"
}
});
// defining dialog control to be used as custom appointment window
var dialog = new ej.Dialog($("#customWindow"), {
width: 600,
height: "auto",
position: { X: 200, Y: 100 },

```

```

showOnInit: false,
enableModal: true,
title: "Appointment Window",
enableResize: false,
allowKeyboardNavigation: false,
close: "clearFields"
});
});
}

```

The following function needs to be defined within script section, which gets called before the default appointment window opens.

### HTML

```

<script type="text/javascript">
function onAppointmentWindowOpen(args) {
args.cancel = true; // prevents the display of default appointment window
var schObj = $("#Schedule1").data("ejSchedule");
// When double clicked on the Scheduler cells, fills the StartTime and
EndTime fields appropriately
$("#StartTime").ejDateTimePicker({ value: args.startTime });
$("#EndTime").ejDateTimePicker({ value: args.endTime });
$("#recWindow").css("display", "none");
$("#appWindow").css("display", "");
if (!ej.isNullOrUndefined(args.target)) {
// When double clicked on the Scheduler cells, if the target is all day
or month cells - only then enable check mark on the all day checkbox
if ($(args.target.currentTarget).hasClass("e-alldaycells") ||
(args.startTime.getHours() == 0 && args.endTime.getHours() == 23))
$("#allDay").prop("checked", true);
else
args.model.currentView == "month" ? $("#allDay").prop("checked", true) :
$("#allDay").prop("checked", false);
// If the target is allDay or month cells - disable the StartTime and
EndTime fields
$("#StartTime,#EndTime").ejDateTimePicker({
enabled: ($(args.target.currentTarget).hasClass("e-alldaycells") ||
(args.startTime.getHours() == 0 && args.endTime.getHours() == 23) ||
$(args.target.currentTarget).hasClass("e-monthcells") ||
args.model.currentView == "month") ? false : true
});
}
// If double clicked on the appointments, fill the custom appointment
window fields with appropriate values.
if (!ej.isNullOrUndefined(args.appointment)) {
$("#customId").val(args.appointment.Id);
$("#subject").val(args.appointment.Subject);
$("#StartTime").ejDateTimePicker({ value: new
Date(args.appointment.StartTime) });
$("#EndTime").ejDateTimePicker({ value: new
Date(args.appointment.EndTime) });
// Fills the Appointment type dropdown with its value
var value = args.appointment.AppointmentType;
$("#AppointmentType").ejDropDownList("clearText");
$("#AppointmentType").ejDropDownList({

```

```

text: value,
value: value
});
$("#allDay").prop("checked", args.appointment.AllDay);
$("#recurrence").ejCheckBox({ checked: args.appointment.Recurrence });
if (args.appointment.Recurrence) {
$("#editor").css("display", "");
$("#recSummary").html(args.appointment.RecurrenceRule);
$("#summaryTr").css("display", "");
recObj._recRule = args.appointment.RecurrenceRule; // app recurrence rule
is stored in Recurrence editor object
recObj.recurrenceRuleSplit(args.appointment.RecurrenceRule,
args.appointment.recurrenceExDate); //splitting the recurrence rule
recObj.showRecurrenceSummary(args.appointment.Id); // updating the
recurrence rule in Recurrence editor
}
}
$("#customWindow").ejDialog("open");
}
</script>

```

On clicking the **Submit** button within the Custom Appointment window, the following function gets executed – which will validate the appointment fields and then save it appropriately.

### HTML

```

<script type="text/javascript">
function save() {
// checks if the subject value is not left blank before saving it.
if ($.trim($("#subject").val()) == "") {
$("#subject").addClass("error");
return false;
}
var obj = {}, temp = {}, rType;
var formElement = $("#customWindow").find("#custom").get(0);
// looping through the custom form elements to get each value and form a
JSON object
for (var index = 0; index < formElement.length; index++) {
var columnName = formElement[index].name, $element =
$(formElement[index]);
if (columnName != undefined) {
if (columnName == "")
columnName = formElement[index].id.replace(this._id, "");
if (columnName != "" && obj[columnName] == null) {
var value = formElement[index].value;
if (columnName == "Id" && value != "")
value = parseInt(value);
if ($element.hasClass("e-datetimepicker"))
value = new Date(value);
if (formElement[index].type == "checkbox")
value = formElement[index].checked;
obj[columnName] = value;
}
}
}
obj["RecurrenceRule"] = (obj.Recurrence) ? recurRule : null;

```

```

var appTypeObj = $("#AppointmentType").data("ejDropDownList");
obj["AppointmentType"] = appTypeObj.getSelectedValue();
$("#customWindow").ejDialog("close");
var object = $("#Schedule1").data("ejSchedule");
object.saveAppointment(obj);
clearFields();
}
// This function executes when the submit/cancel button in the recurrence
editor window is pressed.
function onRecurrenceClick(args) {
if ($ (args.e.currentTarget).attr("id") == "recSubmit") {
recObj = $("#recurrenceEditor").ejRecurrenceEditor('instance');
recObj.closeRecurPublic();
recurRule = recObj._recurRule;
$("#recSummary").html(recurRule);
}
else {
if (($ (args.e.currentTarget).attr("id") == "recCancel")) {
if ($("#recSummary").html() == "") {
$("#editor").css("display", "none");
$("#recurrence").ejCheckBox({ checked: false });
}
else {
$("#recurrence").ejCheckBox({ checked: true });
}
}
}
}
$("#recWindow").css("display", "none");
$("#appWindow").css("display", "");
if ($("#recSummary").html() != "")
$("#summaryTr").css("display", "");
}
// This function executes when the Edit anchor tag in the edit
appointment window is clicked.
function recurrenceRule() {
$("#recWindow").css("display", "");
$("#appWindow").css("display", "none");
}
// Clears all the field values of the custom window after saving
appointments
function clearFields() {
$("#customId").val("");
recObj.clearRecurrenceFields();
$("#subject").val("");
$("#AppointmentType").val("");
$("#recSummary").html("");
$("#summaryTr").css("display", "none");
$("#recurrence").ejCheckBox({ checked: false });
$("#editor").css("display", "none");
$("#StartTime, #EndTime").ejDateTimePicker({ enabled: true });
}
// This function executes when the recurrence checkbox is checked in the
custom appointment window
function recurCheck(args) {
if (args.isInteraction) {
if ($("#recurrence").get(0).checked == true) {
$("#recWindow").css("display", "");

```



```

$("#appWindow").css("display", "none");
$("#editor").css("display", "");
}
else {
$("#recWindow").css("display", "none");
$("#editor").css("display", "none");
$("#recSummary").html("");
$("#summaryTr").css("display", "none");
}
}
}
// This function executes when the All-day checkbox is checked in the
custom appointment window
function allDayCheck() {
// Disables and sets the specific hours to the StartTime and EndTime
fields, when the all-day checkbox is checked
if ($("#allDay").prop("checked")) {
var a = $("#StartTime").data("ejDateTimePicker").model.value;
a.setHours(0, 0, 0);
var b = $("#EndTime").data("ejDateTimePicker").model.value;
b.setHours(23, 59, 0);
$("#StartTime").ejDateTimePicker({
value: new Date(a),
enabled: false
});
$("#EndTime").ejDateTimePicker({
value: new Date(b),
enabled: false
});
} else {
$("#StartTime").ejDateTimePicker({ enabled: true });
$("#EndTime").ejDateTimePicker({ enabled: true });
}
}
// This function executes when the subject text field is currently being
focused
function temp() {
$("#subject").removeClass("error");
}
// This function executes when the cancel button in the custom
appointment window is pressed.
function cancel() {
recObj = $("#recurrenceEditor").ejRecurrenceEditor('instance');
clearFields();
$("#customWindow").ejDialog("close");
}
}
</script>

```

### Scheduler Customization using queryCellInfo

It is possible to format and customize almost every child elements of scheduler such as work cells, header cells, time cells and so on using [queryCellInfo](#) event.

The following code snippet shows how to customize the appointment and work cells based on the query cell info event.

### HTML

```
// Container for ejScheduler widget
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Scheduler($("#Schedule1"), {
queryCellInfo: "checkInfo"
});
});
}
```

### HTML

```
<script type="text/javascript">
function checkInfo(args) {
switch (args.requestType) {
case "workcells":
args.element.css("background-color", "#ffe9cc");
break;
case "monthcells":
args.element.css("background-color", "#faa41a");
args.element.css("border-color", "#faa41a");
break;
}
}
</script>
```

The Scheduler elements are listed below which can be formatted through this event. The names are listed in the format with which it can be accessed or used within the requestType argument of the event.

| Request Type        | Description   |
|---------------------|---|
| appointment         | Depicts the appointment element within the Scheduler.                     |
| agendacells         | Depicts the Agenda Cell element within the Scheduler.                     |
| alldaycells         | Depicts the AllDay cell element within the Scheduler.                     |
| headercells         | Depicts the header cell element within the Scheduler.                     |
| resourceheadercells | Depicts the resource header cell element within the Scheduler.            |
| leftheadcells       | Depicts the left empty space on header cell element within the Scheduler. |
| leftindentcells     | Depicts the left empty space on date cell element within the Scheduler.   |
| timecells           | Depicts the left side time panel cell element within the Scheduler.       |
| headerdate          | Depicts the header date cell element within the Scheduler.                |

| Request Type        | Description   |
|---------------------|---|
| emptytd             | Depicts the empty space above the vertical scroller within the Scheduler. |
| resourcegroupheader | Depicts the header group cell in horizontal orientation in the Scheduler. |
| monthcells          | Depicts the month cell element within the Scheduler.                      |
| workcells           | Depicts the work cell element within the Scheduler.                       |

## Navigation

Navigation in Scheduler can be classified based on Scheduler views, date and also the appointments in it.

### View Navigation

By default, all the available [view options](#) except the Custom View are available at the top right corner of the Schedule header, which can be traverse through continuously as and when needed.

Clicking on the particular date header in the Week/Work Week/Month/Custom View will navigate to the day view automatically. Also, clicking on the week header ranges displayed at the left side in the month view will navigate to the Week view. These particular actions can take place only if the Week and Day view options are present in the [views](#) Collection.

### Handling View navigation actions

Usually, the [navigation](#) event gets triggered whenever the views/dates are being navigated. To block the navigation to day and week views from month view, the **navigation** event can be used in the following way.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
    $(function () {
        var sample = new ej.Schedule($("#Schedule1"), {
            currentDate: new Date(2015, 11, 2),
            appointmentSettings: {
                dataSource: [{
                    Id: 100,
                    Subject: "Research on Sky Miracles",
                    StartTime: new Date(2015, 11, 2, 9, 00),
                    EndTime: new Date(2015, 11, 2, 10, 30)
                }]
            },
            navigation: "onNavigation"
        });
    });
}
```

**JAVASCRIPT**

```
function onNavigation(args) {
    //args.target.currentTarget - target element which is clicked.
    var target = $(args.target.currentTarget);
    if (args.requestType == "viewNavigate" && (target.hasClass("e-
headercells") || target.hasClass("e-monthheader") || target.hasClass("e-
timecells")))
        args.cancel = true;
}
```

**Note:** Based on the navigation, the appointments that lies between the particular date ranges of the current view are fetched and rendered in the Scheduler.

**Date Navigation**

The Scheduler date can be navigated on two aspects either in a continuous or random manner. On pressing the previous and next navigation arrow icons in the Scheduler header will move the scheduler one date back and forth respectively.

Another way of navigating through date is by making use of the built-in calendar available within the Scheduler, which pops out when the header date range is clicked. Selecting any date in the calendar will make the Scheduler to move to that particular date appropriately.

**Handling date navigation actions**

To handle the date navigation actions, the [navigation](#) event can be used. For example, to block the date navigation, follow the below code example.

**HTML**

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

**JAVASCRIPT**

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
    $(function () {
        var sample = new ej.Schedule($("#Schedule1"), {
            currentDate: new Date(2015, 11, 2),
            appointmentSettings: {
                dataSource: [{
                    Id: 100,
                    Subject: "Research on Sky Miracles",
                    StartTime: new Date(2015, 11, 2, 9, 00),
                    EndTime: new Date(2015, 11, 2, 10, 30)
                }]
            },
            navigation: "onNavigation"
        });
    });
}
```

**JAVASCRIPT**

```
function onNavigation(args) {
  //args.target - target element which is clicked.
  //args.currentDate - current date of the Scheduler.
  //args.requestType - Specifies the navigation type.
  if (args.requestType == "dateNavigate")
    args.cancel = true;
}
```

### Appointment Navigation

The Appointment navigation bars (labeled **Previous/Next Appointment**) are rendered parallel to each other on the left and right centric corners of the Scheduler control. It is controlled by an API [showAppointmentNavigator](#) which is set to true by default. When it is set to false, these bars will not be displayed on the Scheduler.

Whenever the previous/Next Appointment bars are clicked, it navigates the Scheduler to the corresponding closest date where the appointments are available. If no appointments are available beyond the current date, then these appointment bars will be in a disabled state.

The following code example shows the way to define the **showAppointmentNavigator** property for Scheduler.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#Schedule1"), {
      currentDate: new Date(2015, 11, 2),
      showAppointmentNavigator: true,
      appointmentSettings: {
        dataSource: [{
          Id: 100,
          Subject: "Research on Sky Miracles",
          StartTime: new Date(2015, 11, 7, 9, 00),
          EndTime: new Date(2015, 11, 7, 10, 30)
        }]
      }
    });
  });
}
```

### Template

Template is applicable to all the below specified elements of the Scheduler,

- Appointments
- Cells
- Resource header

- Date header
- Priority field
- TimeScale
- Date and time columns in Agenda view
- Tooltip

### Appointment Template

The template design that applies on for the Scheduler appointments. The field names that are mapped from the dataSource to the appropriate field properties within the [appointmentSettings](#) can be accessed within the template.

Apart from the dataSource field names, the template can also access the current view of the Scheduler using the name **View** – which can contain either of the following values in lowercase.

- day
- week
- workweek
- agenda
- month
- custom view

It is controlled by an API named [appointmentTemplateId](#) which accepts the id value of the template design block preceded by a symbol #.

Usually, the appointments are displayed with its **Subject** and **Start/End time** on the Scheduler. If suppose, the subject needs to be accompanied with location text, it can be done with the following code example.

#### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<script id="appTemplate" type="text/x-jsrender">
{{"{{"}}"}}if View !== "agenda"{{"{{"}}}}
<div style="height:100%; background-color:orange; margin-left: 5px;">
<div style="margin-left: 2px;">{{"{{"}}":Subject{{"{{"}}}}</div>
<div style="margin-left: 2px;">{{"{{"}}":Location{{"{{"}}}}</div>
</div>
{{"{{"}}}}else{{"{{"}}}}
<div>{{"{{"}}":Subject{{"{{"}}}}, {{"{{"}}":Location{{"{{"}}}}</div>
{{"{{"}}}}/if{{"{{"}}}}
</script>
```

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
appointmentTemplateId: "#appTemplate",
appointmentSettings: {
```

```

dataSource: [{
  Id: 100,
  Subject: "Wild Discovery",
  StartTime: new Date(2015, 11, 2, 9, 00),
  EndTime: new Date(2015, 11, 2, 10, 30),
  Location: "CHINA"
}]
}
});
});
}

```

## Cell Templates

The template design that applies on the Scheduler elements such as all-day cells, work cells and month cells which allows the customization to be done based on the date, view, resources and timescale. The cells can be customized to add images, colors, and other elements etc and can also access the current view of the Scheduler using the name **view**.

**All-day cells** - An API named [allDayCellsTemplateId](#) can be used to customize the all-day cells, which accepts the id of the template design block preceded with a symbol #.

**Work cells and Month cells** - An API named [workCellsTemplateId](#) can be used to customize the work cells in all the views, which accepts the id of the template design block preceded by a symbol #.

The cells can be customized with the following code example.

## HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<!-- Template for All-day cells -->
<script id="allDayTemplate" type="text/x-jsrender">
<div class="e-icon e-scheduleallday" style="opacity:0.5"></div>
<span style="opacity:0.5">AllDay</span>
</script>
<!-- Template for Workcells and Monthcells -->
<script id="workTemplate" type="text/x-jsrender">
{{"{{"}}if resource.classname == 'e-parentnode'{{"{{"}}}}
{{"{{"}}:resource.text{{"{{"}}}}
{{"{{"}}else{{"{{"}}}}
{{"{{"}}if date.getDay() == 0 || date.getDay() == 6{{"{{"}}}}
<div style="background-color:lightblue">Weekend</div>
{{"{{"}}else{{"{{"}}}}
{{"{{"}}if view == 'month' && resource.text == 'Party Hall-A' &&
date.getDay() == 5{{"{{"}}}}
<div style="background-color:burlywood">Meeting</div>
{{"{{"}}else resource.text != 'Party Hall-B' && date.getDate() ==
15{{"{{"}}}}
<div style="background-color:thistle">Holiday</div>
{{"{{"}}else view != 'month' && resource.text == 'Party Hall-A' &&
date.getDay() == 5 && date.getHours() == 10{{"{{"}}}}
<div style="background-color:burlywood">Meeting</div>
{{"{{"}}else view == 'month' && resource.text == 'Party Hall-B' &&
date.getDay() == 5{{"{{"}}}}
<div style="background-color:lightblue">Conf.</div>

```

```

{{"{{"}}else resource.text == 'Party Hall-B' && date.getDate() ==
16{{}}}
<div style="background-color:darkkhaki">HappyDay</div>
{{"{{"}}else view != 'month' && resource.text == 'Party Hall-B' &&
date.getDay() == 5 && date.getHours() == 12{{}}}
<div style="background-color:goldenrod">Conf.</div>
{{"{{"}}else date.getDate() == 10 && date.getMonth() == 11{{}}}
<div style="background-color:palegreen">Day Special</div>
{{"{{"}}else date.getDate() == 25 && date.getMonth() == 11{{}}}
<div style="background-color:sandybrown">Christmas</div>
{{"{{"}}/if{{}}}
{{"{{"}}/if{{}}}
{{"{{"}}/if{{}}}
</script>

```

## JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
allDayCellsTemplateId: "#allDayTemplate",
workCellsTemplateId: "#workTemplate",
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});
}
}

```

## Date Header Template

The template design that applies on for the date header part of the Scheduler. An API named [dateHeaderTemplateId](#) can be used to customize the date header which accepts the id value of the template design block preceded by a symbol #. The template can also access the current view of the Scheduler in using the name **view**.

The Date header can be customized with the following code example.

## HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<!-- Template for Dateheader -->
<script id="dateTemplate" type="text/x-jsrender">
<div>{{"{{"}}:~dTemplate(date) {{}}}{{}}</div>
</script>

```



**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
dateHeaderTemplateId: "#dateTemplate",
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});
}

```

**JAVASCRIPT**

```

function _dateFormat(date) {
var dFormat = ej.format(new Date(date), "dd/MM");
return dFormat;
}
$.views.helpers({ dTemplate: _dateFormat });

```

**Resource Header Template**

The template structure that applies on the resource headers of the Scheduler. By default, only the resource names will be displayed on the resource header bar. Also, the way of rendering resource headers on the Scheduler is comparatively different for both the vertical and horizontal scheduler views.

The field names that are mapped from the dataSource to the appropriate field properties within the **resourceSettings** can be accessed within the resource header template.

**Resource Header Template in Vertical View**

To customize the resource header with some additional images or other customizations in **vertical Scheduler View** – refer the below code example.

**HTML**

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<script id="resTemplate" type="text/x-jsrender">
<div style="height:100%">
<div style="width:15px;height:15px;margin-left:275px;margin-
top:2px;float:left;background:{ "{":""}:ResourceColor{}}{}";"></div><div
style="float:left;margin-left:5px;">{{ "{":""}:ResourceText{}}{}"></div>
</div>
</script>

```

**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
currentDate: new Date(2015, 04, 05),
resourceHeaderTemplateId: "#resTemplate",
group: {
resources: ["Rooms"]
},
resources: [{
field: "roomId",
title: "Room",
name: "Rooms",
allowMultiple: false,
resourceSettings: {
dataSource: [{
ResourceText: "ROOM1",
id: 1,
ResourceColor: "orange"
}, {
ResourceText: "ROOM2",
id: 2,
ResourceColor: "#56ca85"
}],
text: "ResourceText",
id: "id",
color: "ResourceColor"
}
}],
appointmentSettings: {
resourceFields: "roomId",
dataSource: [{
Id: 101,
Subject: "Talk with Nature",
StartTime: new Date(2015, 11, 5, 10, 00),
EndTime: new Date(2015, 11, 5, 11, 00),
roomId: 2
}]
}
});
});
}

```

#### Resource Header Template in Horizontal View

To perform the above specified same customization in **horizontal Scheduler view**, the template structure varies a little bit as depicted below.

#### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<script id="resTemplate" type="text/x-jsrender">
<div style="height:100%">

```

```
<div style="width:15px;height:15px;margin-right:5px;margin-top:2px;float:left;background:{{"{{"}}:ResColor{{}}}};"></div><div>{{"{{"}}:ResText{{}}}}</div>
</div>
</script>
```

## JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
height: "500px",
currentDate: new Date(2015, 04, 05),
orientation: "horizontal",
resourceHeaderTemplateId: "#resTemplate",
group: {
resources: ["Rooms"]
},
resources: [{
field: "roomId",
title: "Room",
name: "Rooms",
allowMultiple: false,
resourceSettings: {
dataSource: [{
ResText: "ROOM1",
id: 1,
ResColor: "orange"
}, {
ResText: "ROOM2",
id: 2,
ResColor: "#56ca85"
}],
text: "ResText",
id: "id",
color: "ResColor"
}
}],
appointmentSettings: {
resourceFields: "roomId",
dataSource: [{
Id: 101,
Subject: "Talk with Nature",
StartTime: new Date(2015, 11, 5, 10, 00),
EndTime: new Date(2015, 11, 5, 11, 00),
roomId: 2
}]
}
});
});
}
```

**Note:** In horizontal Scheduler, the header template makes use of an additional field namely **classname** which holds either **e-parentnode** or **e-childnode** value. The field **classname** can be used in the application scenario to check for the parent or child header node. You can apply the different template customization accordingly based on it.

### TimeScale Templates

The [TimeScale](#) is also availed with template options to allow customization. It includes the following 2 properties for customization -

- [majorSlotTemplateId](#) - Accepts the id value of the template design block preceded by a symbol #, which gets applied for the major time slots.
- [minorSlotTemplateId](#) - Accepts the id value of the template design block preceded by a symbol #, which gets applied for the minor time slots.

The template customization for major and minor timeslots can be referred from the following code example.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<!-- Template for Majorslot -->
<script id="majorTemplate" type="text/x-jsrender">
<div>{{"{{"}}}:~major(date){{}}}}</div>
</script>
<!-- Template for Minorslot -->
<script id="minorTemplate" type="text/x-jsrender">
<div>{{"{{"}}}:~minor(date){{}}}}</div>
</script>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
timeScale: {
enable: true,
majorSlot: 60,
majorSlotTemplateId: "#majorTemplate",
minorSlotCount: 6,
minorSlotTemplateId: "#minorTemplate"
},
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
}
```

```
});
});
}
```

## JAVASCRIPT

```
function _majorFormat(date) {
var dFormat = ej.format(new Date(date), "hh:mm:ss");
return dFormat;
}
function _minorFormat(date) {
var dFormat = ej.format(new Date(date), "hh:mm:ss");
return dFormat;
}
$.views.helpers({
major: _majorFormat,
minor: _minorFormat
});
```

## Priority Settings Template

The template design which can be applied to the content of the priority field in the appointment window. By default, the appropriate icons are displayed for each priority options such as **None**, **High**, **Medium** and **Low**.

When template is applied for the [prioritySettings](#), these default icons will be replaced by the custom icons or styles defined newly. The following code example depicts the way to enable the priority settings and to define the new custom styles to replace the default icons in the Priority field.

## HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<style type="text/css">
.critical,
.ultra-critical,
.none {
height: 13px;
width: 13px;
float: left;
margin-right: 4px;
background-repeat: no-repeat;
background-size: 60px;
padding: 1px;
margin-top: 2px;
}
.critical {
background-color: orange;
background-position: -13px;
}
.ultra-critical {
background-color: #56ca85;
background-position: -59px;
}
</style>
```

**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
prioritySettings: {
enable: true,
template: "<div class='${value}'></div>",
dataSource: [{
text: "None",
id: 1,
value: "none"
}, {
text: "Critical",
id: 2,
value: "critical"
}, {
text: "Ultra Critical",
id: 3,
value: "ultra-critical"
}]
},
appointmentSettings: {
priority: "Priority",
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA",
Priority: "critical"
}]
}
});
});
}

```

The custom style class names defined for the priority template should be same as that of the values defined for each priority option within the dataSource, so that it applies properly.

**Note:** Additionally, the priority field within the [appointmentSettings](#) should be defined with appropriate dataSource field name. When an appointment is assigned with a priority value, the custom style/icon defined for that priority option will get applied over that appointment.

**Tooltip Template**

The tooltip can be applied with the customized template design. Currently the tooltip support is provided only for the appointments and the default tooltip displays the Subject and duration on hovering across the appointments.

By making use of template feature with tooltip, all the field names that are mapped from the dataSource to the appropriate field properties within the **appointmentSettings** can be accessed.

To define the template option for tooltip, the [tooltipSettings](#) must be enabled first. The following code example depicts the way to add the tooltip template.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<script id="tooltipTemplate" type="text/x-jsrender">
<div style="width:145px">
<div style="padding-top:3px;">
<div style="float:left; font:13px Segoe UI; font-
weight:bold;">Subject:&#160;&#160;:&#160;</div>
<div style="padding-top:2px; font:12px Segoe UI
SemiBold;">{{"{{"}}:Subject{{}}}}</div>
</div>
<div style="padding-top:3px">
<div style="float:left; font:13px Segoe UI; font-
weight:bold;">Location:&#160;</div>
<div style="padding-top:2px; font:12px Segoe UI
SemiBold;">{{"{{"}}:Location{{}}}}</div>
</div>
</div>
</script>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
tooltipSettings: {
enable: true,
templateId: "#tooltipTemplate"
},
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});
}
```

### Agenda View Templates

Agenda View provides two separate templates – one for date column and another for time column. These templates allows the customization of the content of both the date and time columns. Apart from this, the event column can also be customized through the existing API named [appointmentTemplateId](#).

The following code snippet shows how to customize the content of the date, time and event column.

**HTML**

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
// Template for date column
<script id="dateTemplate" type="text/x-jsrender">
<div style="height:100%">
<div>
<div>{{"{{"}}:~dateDisplay(StartTime){}}}</div>
</div>
</div>
</script>
// Template for time column
<script id="timeTemplate" type="text/x-jsrender">
<div style="height:100%">
<div>
<div>{{"{{"}}:~timeDisplay(StartTime){}}}</div>
</div>
</div>
</script>
// Template for appointment which applies for event column in agenda
view.
<script id="appTemplate" type="text/x-jsrender">
{{"{{"}}if View !== "agenda"{{}}}}
<div style="height:100%; background-color:orange; margin-left: 5px;">
<div style="margin-left: 2px;">{{"{{"}}:Subject{{}}}</div>
<div style="margin-left: 2px;">{{"{{"}}:Location{{}}}</div>
</div>
{{"{{"}}else{{}}}}
<div>{{"{{"}}:Subject{{}}}, {{"{{"}}:Location{{}}}</div>
{{"{{"}}/if{{}}}}
</script>

```

**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
appointmentTemplateId: "#appTemplate",
agendaViewSettings: {
dateColumnTemplateId: "#dateTemplate",
timeColumnTemplateId: "#timeTemplate"
},
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});

```



```
});
}
```

## JAVASCRIPT

```
function _getDate(date) {
var dateCol = new Date(date);
return dateCol.toString();
}
function _getTime(date) {
var time = new Date(date);
return time.toLocaleTimeString();
}
//Here, used the helper function to get the date and time value part from
the StartTime.
$.views.helpers({
dateDisplay: _getDate,
timeDisplay: _getTime
});
```

## Globalization and Localization

### Globalization

The Scheduler control is built with default **globalization** support as it format the dates according to the user's locale automatically and processes it internally without any need for manual conversions. This kind of default handling of Scheduler dates is achieved through the built-in **ej.globalize** library which globalize the date, day and month names accordingly.

### Localization

Scheduler also comes with default localization support which allows it to customize the display of text within the Scheduler in a user-specific culture and locale. The Schedule control can be localized in specific culture using the common API [locale](#) along with the collection of localized words defined for that culture using the `ej.Schedule.Locale [culture-code]`.

---

**Note:** By default, the Schedule control is localized in **en-US** culture.

---

To localize Scheduler into any particular culture, it is necessary to refer the culture-specific script files in your application after the reference of **ej.web.all.min.js** file, which are available under the following location.

<Installed location>\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

The following code example shows how to localize the Schedule control in **fr-FR** culture.

## HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

## JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
```

```

ej.Schedule.Locale["fr-FR"] = {
  ReminderWindowTitle: "Fenêtre de rappel",
  CreateAppointmentTitle: "créer un rendez-",
  RecurrenceEditTitle: "Modifier répétition nomination",
  RecurrenceEditMessage: "Comment voulez-vous changer le cas dans la
  série?",
  RecurrenceEditOnly: "Seulement cette nomination",
  RecurrenceEditSeries: "La série entière",
  PreviousAppointment: "Nomination précédente",
  NextAppointment: "prochain rendez-vous",
  AppointmentSubject: "sujet",
  StartTime: "Heure de début",
  EndTime: "Heure de fin",
  AllDay: "toute la journée",
  Today: "aujourd'hui",
  Recurrence: "répétition",
  Done: "Terminé",
  Cancel: "annuler",
  Ok: "Ok",
  RepeatBy: "Répétez par",
  RepeatEvery: "répéter chaque",
  RepeatOn: "répéter l'opération sur",
  StartsOn: "démarre sur",
  Ends: "extrémités",
  Summary: "résumé",
  Daily: "quotidien",
  Weekly: "hebdomadaire",
  Monthly: "mensuel",
  Yearly: "annuel",
  Every: "tous",
  EveryWeekDay: "chaque jour de la semaine",
  Never: "jamais",
  After: "après",
  Occurrence: "apparition",
  On: "sur",
  Edit: "Modifier",
  RecurrenceDay: "Jour (s)",
  RecurrenceWeek: "Semaine (s)",
  RecurrenceMonth: "Mois (s)",
  RecurrenceYear: "Année (s)",
  The: "la",
  OfEvery: "de chaque",
  First: "première",
  Second: "deuxième",
  Third: "troisième",
  Fourth: "quatrième",
  Last: "dernier",
  WeekDay: "jour de la semaine",
  WeekEndDay: "Jour de week-end",
  Subject: "sujet",
  Categorize: "Catégories",
  DueIn: "En raison",
  DismissAll: "rejeter tout",
  Dismiss: "rejeter",
  OpenItem: "Ouvrir l'élément",
  Snooze: "répétition",
  Day: "jour",

```

```

Week: "semaine",
WorkWeek: "Semaine de travail",
Month: "mois",
AddEvent: "Ajouter événement",
CustomView: "Vue personnalisée",
Agenda: "ordre du jour",
Detailed: "détaillé",
EventBeginsin: "Nomination commence dans",
Editevent: "Modifier nomination",
Editseries: "Modifier série",
Times: "fois",
Until: "jusqu'à",
Eventwas: "rendez-vous était",
Hours: "hrs",
Minutes: "minutes",
Overdue: "en retard",
Days: "jour (s)",
Event: "Sujet",
Select: "sélectionner",
Previous: "prev",
Next: "suivant",
Close: "proche",
Delete: "effacer",
Date: "date",
Showin: "montrer en",
Gotodate: "Aller à la date",
Resources: "RESSOURCES",
RecurrenceDeleteTitle: "Supprimer répétition rendez-",
Location: "emplacement",
Priority: "priorité",
RecurrenceAlert: "alerte",
WrongPattern: "Le modèle de récurrence est pas valable",
CreateError: "La durée de la nomination doit être plus courte que la
façon dont elle se produit fréquemment. Raccourcir la durée ou changer le
modèle de récurrence dans la boîte de dialogue Récurrence de rendez.",
DragResizeError: "Impossible de replanifier une occurrence du rendez-vous
récurrent. si elle saute sur une occurrence ultérieure du même rendez-
vous.",
StartEndError: "L'heure de fin doit être supérieur à l'heure de début",
MouseOverDeleteTitle: "supprimer un rendez-",
DeleteConfirmation: "Êtes-vous sûr de vouloir supprimer ce rendez-vous?",
Time: "Temps"
};
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
locale: "fr-FR",
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});

```

```
}

```

**Note:** Refer the **ej.culture.fr-FR.min.js** file in your HTML application and also define the **locale** property for the Schedule control with the appropriate **culture-code [fr-FR]**.

For further information on – how to refer the required culture scripts into your application, refer [here](#).

### Localizing Specific Words

To customize or localize only some specific words in the default `ej.Schedule.Locale["en-US"]` collection, the words to be localized/customized can be defined in a separate variable and then extended to the original collection as depicted in the following code example.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var customizationMessage = {
// customize the appointment window title
CreateAppointmentTitle: "Create Event",
// customize the view options text in the Schedule header
Day: "1 Day",
Week: "7 Days",
WorkWeek: "5 Days",
Month: "Month"
};
// Extend only the required changes to the original locale collection
$.extend(ej.Schedule.Locale["en-US"], customizationMessage);
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
height: "525px",
currentDate: new Date(2015, 11, 5),
appointmentSettings: {
dataSource: [{
Id: 101,
Subject: "Talk with Nature",
StartTime: new Date(2015, 11, 5, 10, 00),
EndTime: new Date(2015, 11, 5, 11, 00)
}]
}
});
});
}

```

### Time Zone

The Scheduler makes use of the System time zone by default. If it needs to follow some other user-specific time zone, then the API [timeZone](#) can be used. Also, the Scheduler can be set to observe the Daylight Saving Time (DST) with its **isDST** property which is set to **false** by default.

When [isDST](#) property is set to **true**, the Scheduler internally processes the time difference values (for the Start and end time of the appointments) related to the Scheduler time zone that observes daylight savings time.

The following code example shows the way to set the specific time zone value with the daylight savings time observed in the Scheduler.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
timeZone: "UTC +05:30",
isDST: true,
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});
}
```

#### *Setting different TimeZone for Scheduler Appointments*

Apart from the default action of applying specific timezone to the entire Scheduler, it is also possible to set different time zone values for each appointments through the properties **startTimeZone** and **endTimeZone** which can be defined as separate fields within the appointment dataSource. When these properties are not explicitly defined for appointments, the appointments Start and End time will be processed based on the Scheduler time zone.

**Note:** The **isDST** property closely relies on the appointment fields like [StartTimeZone](#) and [EndTimeZone](#), for appropriate time difference calculations. If these two fields are not defined for appointments, then **isDST** depends on the System **timeZone** value.

The following code snippet shows how to define isDST and the time zones for specific appointments.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
isDST: true,
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA",
StartTimeZone: "UTC +02:00",
EndTimeZone: "UTC +02:00"
}]
}
});
});
}

```

### Customizing the TimeZone Collection

It is also possible to define or customize the default time zone collection of the Scheduler, by using the [timeZoneCollection](#) API as follows.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
timeZoneCollection: {
dataSource: [
{ text: "UTC -04:00", id: "10", value: "UTC -04:00" },
{ text: "UTC -03:30", id: "11", value: "UTC -03:30" },
{ text: "UTC -03:00", id: "12", value: "UTC -03:00" },
{ text: "UTC -02:00", id: "13", value: "UTC -02:00" },
{ text: "UTC -01:00", id: "14", value: "UTC -01:00" },
{ text: "UTC +00:00", id: "15", value: "UTC +00:00" },
{ text: "UTC +01:00", id: "16", value: "UTC +01:00" },
{ text: "UTC +02:00", id: "17", value: "UTC +02:00" },
{ text: "UTC +03:00", id: "18", value: "UTC +03:00" },
{ text: "UTC +03:30", id: "19", value: "UTC +03:30" },
{ text: "UTC +04:00", id: "20", value: "UTC +04:00" },
{ text: "UTC +04:30", id: "21", value: "UTC +04:30" },
{ text: "UTC +05:00", id: "22", value: "UTC +05:00" }
],
}
}
});
});
}

```

```

text: "text",
id: "id",
value: "value",
},
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA",
StartTimeZone: "UTC +02:00",
EndTimeZone: "UTC +02:00"
}]
}
});
});
}

```

**Note:** The values defined within the **timeZoneCollection** dataSource are usually the only options displayed within the start and end time zone dropdown fields of the appointment window.

### Time Mode

The time mode of the Scheduler can be either **12** or **24 hours** format which is based on the [locale](#) set to the Scheduler. Since the default locale value of the Scheduler is **en-US**, therefore the time mode will be set to **12 hours** format (by default) automatically based on the culture.

The user can also set specific time mode for the Scheduler using [timeMode](#) property which accepts either **String** or **enum** value. It accepts the following **enum** values,

- ej.Schedule.TimeMode.Hour12
- ej.Schedule.TimeMode.Hour24

The following code snippet shows the way to set specific **24 hour format** time mode for the Scheduler.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<script type="text/javascript">
$(function() {
$("#Schedule1").ejSchedule({
currentDate: new Date(2015, 11, 2),
timeMode: ej.Schedule.TimeMode.Hour24,
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});

```

```
</script>
```

**Note:** If the **timeMode** property is not set with specific value, then the value will be taken based on the locale assigned for the Scheduler.

### Date Format

Scheduler can be used with all valid date formats. The default date format used in Scheduler is "MM/dd/yyyy".

If the [dateFormat](#) property is not specified particularly, then it will be taken based on the locale that is assigned to the Scheduler. The default locale applied on the Scheduler is "en-US", which makes it to follow the "MM/dd/yyyy" pattern by default.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#Schedule1"), {
      currentDate: new Date(2015, 11, 5),
      dateFormat: "yyyy/MM/dd",
      appointmentSettings: {
        dataSource: [{
          Id: 101,
          Subject: "Talk with Nature",
          StartTime: new Date(2015, 11, 5, 10, 00),
          EndTime: new Date(2015, 11, 5, 11, 00)
        }]
      }
    });
  });
}
```

### First Day of Week

The [firstDayOfWeek](#) property allows to set any of the week days as start of the week/workweek/month view in Scheduler. It accepts either the **integer** (Sunday=0, Monday=1, Tuesday=2, etc) or **string** ("Sunday", "Monday", etc) or **ej.Schedule.DayOfWeek** enum type value. The default value of this **firstDayOfWeek** depends on the current culture (language) assigned to the Scheduler.

To set different first day of week in Scheduler,

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT



```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
// Set the Active view
currentView: ej.Schedule.CurrentView.Week,
// Configure the week start day(First day of week)
firstDayOfWeek: ej.Schedule.FirstDayOfWeek.Tuesday,
currentDate: new Date(2015, 11, 7),
appointmentSettings: {
//Array of JSON data configure in dataSource
dataSource: [{
Id: 1,
Subject: "Music Class",
StartTime: new Date("2015/11/7 06:00 AM"),
EndTime: new Date("2015/11/7 07:00 AM")
}, {
Id: 2,
Subject: "School",
StartTime: new Date("2015/11/7 9:00 AM"),
EndTime: new Date("2015/11/7 02:30 PM")
}]
}
});
});
}

```

**Note:** The sub-control datepicker which is used within Scheduler for start/end time fields in appointment window and for date navigation purposes will also follow the same first day of week.

### Keyboard Navigation

The shortcut keys for accessing the sub-elements of the scheduler and other Scheduler actions are tabulated in the following table.

| Keys             | Functionality description   |
|------------------|---|
| arrow keys       | To traverse through the Scheduler cells.  |
| Shift+arrow keys | Multiple cell selection   |
| Enter            | Pressing enter key, after a single/multiple scheduler cell selection will make the quick appointment window to pop-up.<br><br>Also, when the focus is being moved on to the fields like checkbox or buttons of the appointment window, pressing enter will select that particular action. |
| Esc              | Closes any of the popup that displays on the Schedule control.  |
| Alt+N            | Opens the Create Appointment window   |
| Ctrl+E           | Opens the Edit Appointment window   |
| Alt+C            | Opens the calendar popup within the Scheduler.  |

|                  |   |
|------------------|---|
| Ctrl+left arrow  | Previous date navigation  |
| Ctrl+right arrow | Next date navigation  |
| Alt++            | Forward traversing of view items in the toolbar   |
| Alt+-            | Reverse traversing of view items in the toolbar   |
| Space            | When the previous/next navigation icons are currently being focused, pressing space navigates through the corresponding dates.<br><br>Also, when the focus is being moved on to the fields like checkbox or buttons of the appointment window, pressing enter will select that particular action. |
| Del              | Deletes the currently selected appointment.   |
| Tab              | Traversing through the appointments in a forward direction.   |
| Shift+tab        | Traversal of appointments in a reverse order.   |

**Note:** By default [allowKeyboardNavigation](#) property is set to **true**, which allows the Scheduler to be accessed through the above specified keys.

## Setting Dimension

The dimension normally refers to the height and width of the element. With Scheduler, it is possible to set 2 kinds of dimensions.

- Scheduler dimension (Scheduler control height and width)
- Cell dimension (Scheduler cells height and width)

## Scheduler Dimension

The [height](#) and [width](#) properties can be defined to set the outer dimension of the Scheduler control.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
    $(function () {
        var sample = new ej.Schedule($("#Schedule1"), {
            //Setting dimension of Scheduler
            width: "70%",
            height: "500px"
        });
    });
}
```

---

**Note:** The height and width properties accepts both **pixel** and **percentage** values.

---

## Scheduler Cell Dimensions

### Cell Height

The [cellHeight](#) property allows the Scheduler to set the height of the cells in pixels. The appointment height in vertical mode changes accordingly as per the cell size within which it renders.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
cellHeight: "40px",
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30)
}]
}
});
});
}
```

---

**Note:** In **desktop** mode, the default height value of the cells is set to **20px**, whereas for **mobile** mode – the Scheduler cells are rendered with **40px** by default.

---

### Cell Auto-Height

The height of the cells specifically in timeline view can be made to adjust automatically based on its exceeding appointment count. It is controlled by an API named [showOverflowButton](#) which accepts true or false value, denoting whether to enable/disable the cell auto-height adjusting option. To enable this option, set the value of [showOverflowButton](#) as **false** whereas its default value is **true**.

In **Vertical** view, the same functionality is made applicable only in the **Month View** whereas in **Horizontal** mode, it is applicable in all the views.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
```

```
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
height: "525px",
currentDate: new Date(2014, 4, 5),
currentView: "month",
showOverflowButton: false,
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});
}
```

### Cell Width

The [cellWidth](#) property allows the Scheduler to set the width of the cells in pixels. The appointment width adjusts based on the cell width of the Scheduler.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
cellWidth: "97px",
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30)
}]
}
});
});
}
```

**Note:** When the **cellHeight** and **cellWidth** properties are set with some specific pixel values, the cell size does not adapt to the responsive behavior of the Scheduler while resizing it in desktop/mobile mode.

## Responsiveness

Scheduler is provided with default **responsive** support, so that it adjust or auto-resize it's UI content based on the window or the container size on which it is placed.

### Auto-Resizing Scheduler

By default, setting 100% width to the Scheduler makes it adaptable to the window or its parent container, as and when the browser is resized. This will not allow the scheduler to adapt height-wise.

#### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
width: "100%",
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});
}
```

To auto-resize the Scheduler height – set the **height** property of the Scheduler to **100%** and also set some specific height (either percentage or pixel values) to its parent container (or) to the HTML and body tag elements as shown below.

#### HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" style="height:100%">
<head>
<title>My first HTML page</title>
<!-- required CSS REFERENCES -->
<!-- required SCRIPT REFERENCES -->
</head>
<body style="height:100%">
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
</body>
</html>
```

**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
height: "100%",
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});
}

```

**Note:** When the Scheduler width is set to have **less than 600px**, then the View selection options will be displayed in a **navigation drawer**.

Also, whenever the Scheduler resizes - the width of the work cells adjusts automatically (if no **cellWidth** property is specified with pixel values), but the height of the cells are kept to remain as same as 20px until **cellHeight** property is set explicitly with some pixel values.

**Scheduler in Mobile/Tablets**

The Scheduler layout adapts automatically in the mobile and tablet devices with the appropriate UI changes, as the [isResponsive](#) property is set to **true** by default. In case, if the user wants to display the normal scheduler in mobile devices without any adaptive enhancements, then the **isResponsive** property can be set to false.

Some of the default changes done for adaptive Scheduler to render in mobile devices are as follows,

- Animation effect introduced between view/date navigation.
- Cell Height increased
- The header date display changes (displayed next to the navigation icons).
- View options displayed in Navigation drawer
- Time cell width decreased
- Quick window display blocked on cell/appointment single click.
- Appointment resizing action blocked.
- Multiple cell selection blocked.
- Delete appointment option made available within the appointment window.
- Week header display in month view hidden.
- With Multiple resources - only one resource is viewable initially on the screen and to further look onto other resources, user need to swipe the screen.

**Note:** To make the Scheduler control to react as responsive in mobile/tablet devices, it is necessary to refer the additional [ej.responsive.css](http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/responsive-css/ej.responsive.css) file in the application.

The following code snippet depicts the Scheduler code with responsive set to true.

### HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" style="height:100%">
<head>
<title>My first HTML page</title>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />
<link href=" http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/responsive-css/ej.responsive.css" rel="stylesheet" />
<!-- Other required SCRIPT REFERENCES -->
</head>
<body style="height:100%">
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
</body>
</html>
```

### JAVASCRIPT

```
/// <reference path="../../../tsfiles/jquery.d.ts" />
/// <reference path="../../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
isResponsive: true,
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});
}
```

### Persistence

State persistence allows the Scheduler to retain the current model value in the browser cookies for state maintenance. This action is handled through the property [enablePersistence](#) which is set to false by default.

When it is set to **true**, some of the Schedule control model values will be retained even after refreshing the page which are listed below.

|                            |                                 |
|----------------------------|---------------------------------|
| currentView                | timeMode                        |
| firstDayOfWeek             | dateFormat                      |
| isDST                      | timeZone                        |
| timeScale                  | startHour                       |
| endHour                    | workHours                       |
| Height                     | Width                           |
| cellHeight                 | cellWidth                       |
| currentDate                | minDate                         |
| maxDate                    | renderDates                     |
| orientation                | views                           |
| workWeek                   | agendaViewSettings.daysInAgenda |
| enableLoadOnDemand         | showLocationField               |
| showAllDayRow              | isResponsive                    |
| enableRecurrenceValidation | showOverflowButton              |
| allowDragAndDrop           | showDeleteConfirmationDialog    |
| showNextPrevMonth          | appointmentDragArea             |

The Schedule properties that are not retained while maintaining state persistence are included within the **ignoreOnPersist** list, which makes it not to persist by default.

## Export and Print

### Export Appointments to ICS file

The Appointments can be exported as a whole collection or else a single appointment alone can be exported from the Scheduler. By default, the appointments are exported to .ics format which can then be imported and used in any of the other external calendars.

#### Single Appointment Exporting

A single appointment can be exported by making use of its Id. You can achieve this functionality by making use of an [exportSchedule](#) method by passing the id of an appointment to export as one of its parameter.

It can also be achieved in another way by enabling the context menu settings and then adding a custom menu option for export appointment functionality. When right clicked on an appointment, and **export Appointment** option is chosen, the exporting functionality can be handled through the [menuItemClick](#) event, within which the **exportSchedule** method should be defined with the following parameters,

- Action name (to be called in the server-side)
- Server Event (optional)
- Appointment Id (mandatory for single Appointment exporting)



The following code example shows the way to export single appointment from the Scheduler.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
contextMenuSettings: {
enable: true,
menuItems: {
appointment: [
{ id: "open", text: "Open Appointment" },
{ id: "delete", text: "Delete Appointment" },
{ id: "export", text: "Export Appointment" }
]
}
},
menuItemClick: "onMenuItemClick",
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});
}
```

### JAVASCRIPT

```
function onMenuItemClick(args) {
if (args.events.ID == "export") {
var obj = $("#Schedule1").data("ejScheduler");
// exportSchedule() method will send a post to the server-side to call a
specified action.
obj.exportSchedule("ExportToICS", null, args.targetInfo.Id);
}
}
```

**Note:** The Id value of the appointment passed in the above code example can be retrieved in the server-side action through Request.Form["AppointmentId"], as the id passed from the script is stored as hidden value in the input field of the form under this name internally.

### Exporting all Appointments

To export the entire Scheduler appointments, the same [exportSchedule](#) method can be used without passing the id value to its parameter list. To achieve this, keep an individual button to export, and when it is clicked - the Scheduler can be allowed to export all the appointments.

The following code example depicts the way to export all the Scheduler appointments as a whole.

#### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<!-- Button div for Export Option-->
<button id="exportBtn">Export</button>
```

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var button = new ej.Button($("#exportBtn"), {
width: "70px",
height: "30px",
click: "onClick"
});
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});
}
```

#### JAVASCRIPT

```
function onClick(args) {
var obj = $("#Schedule1").data("ejSchedule");
// Calls the server-side action ExportToICS
obj.exportSchedule("ExportToICS", null, null);
}
```

The server-side action **ExportToICS** contains the following code example to export the Scheduler appointments.

#### C#

```
public void ExportToICS(FormCollection form)
{
```

```

IEnumerable data;
string JSONModel = Request.Form["ScheduleModel"];
var model = JsonConvert.DeserializeObject<Dictionary<string,
object>>>(JSONModel);
var Id = Request.Form["AppointmentId"];
if (Id != null)
data = db.DefaultSchedules.Where(app => app.Id.ToString() ==
Id.ToString()).ToList(); // To export single appointment
else
data = db.DefaultSchedules.ToList(); // To export all the appointments
ScheduleExport obj = new ScheduleExport(model, data);
}

```

### PDF Export

Scheduler supports exporting it along with all its appointments in PDF format, for which the same [exportSchedule](#) method can be used without passing any id value to its parameter list. To achieve this, keep an individual button to export and when it is clicked, the Scheduler with appointments can be allowed to export as PDF.

The following code example depicts the way to export the Scheduler with appointments in PDF format.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<!-- Button div for Export Option-->
<button id="btnExport">Export</button>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var button = new ej.Button($("#btnExport"), {
width: "80px",
height: "30px",
click: "onExportClick"
});
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
height: "525px",
currentDate: new Date(2014, 4, 5),
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});
}

```

## JAVASCRIPT

```
function onExportClick(e) {  
var obj = $("#Schedule1").data("ejSchedule");  
obj.exportSchedule("http://js.syncfusion.com/ejservices/api/Schedule/PdfExport", null, null);  
e.cancel = true;  
}
```

The server-side action **PDFExport** contains the following code example to export the Scheduler.

## C#

```
public ActionResult PDFExport(string scheduleModel)  
{  
    PdfExport exp = new PdfExport();  
    JavaScriptSerializer serializer = new JavaScriptSerializer();  
    SchedulePDFExport convert = new SchedulePDFExport();  
    // Here calling the public method to convert the model values to  
    // ScheduleProperties type from string type  
    ScheduleProperties scheduleObject =  
        convert.ScheduleSerializeModel(scheduleModel);  
    // Here converting the schedule appointments data into enumerable object  
    // by deserializing  
    IEnumerable scheduleAppointments =  
        (IEnumerable)serializer.Deserialize(Request.Form["ScheduleProcesedApps"],  
        typeof(IEnumerable));  
    // Here passing the theme values from enum collection  
    PdfDocument document = exp.Export(scheduleObject, scheduleAppointments,  
        ExportTheme.FlatSaffron, Request.Form["locale"]);  
    document.Save("Schedule.pdf", HttpContext.ApplicationInstance.Response,  
        HttpReadType.Save);  
    return RedirectToAction("PDFExportController");  
}
```

**Note:** To pass the theme value as a string instead of the enum value, refer to the following code example of passing the string type theme value.

## C#

```
PdfDocument document = exp.Export(scheduleObject, scheduleAppointments,  
    "flat-saffron", Request.Form["locale"]);
```

### Setting Page Options

It is possible to set/change the PDF page settings such as size, margins (left, right, top and bottom), transition, and rotate angle and header/footer values by passing the page settings and page document template object values into the export method.

### Page Settings

You can set/apply the following page property values to the PDF file before exporting it with the Scheduler by using the **PdfPageSettings** object.

#### a) Page Size

b) Orientation

c) Margins (Left, Right, Top, Bottom)

d) Transition (Direction, Dimension, Duration, Motion, PageDuration, Scale, Style)

e) Rotate (RotateAngle0, RotateAngle90, RotateAngle180, RotateAngle270)

f) Height

g) Width

To apply the above page setting values, you need to create an object for the PdfPageSettings class. Then, you can assign the values to the available properties within it such as Size = PdfPageSize.A3, Orientation = PdfPageOrientation.Landscape and so on. Once done, pass this object into the export method which will set these page settings values to the PDF file before exporting it. Refer to the following code example (server-side) to set/change the page settings values.

### C#

```
public ActionResult PDFExport(string scheduleModel)
{
    JavaScriptSerializer serializer = new JavaScriptSerializer();
    SchedulePDFExport convert = new SchedulePDFExport();
    ScheduleProperties scheduleObject =
        convert.ScheduleSerializeModel(scheduleModel); // Here calling the public
method to convert the model values to ScheduleProperties type from string
type
    IEnumerable scheduleAppointments =
        (IEnumerable)serializer.Deserialize(Request.Form["ScheduleProcesedApps"],
        typeof(IEnumerable)); // Here deserialize the appointments to enumerable
object
    PdfExport exp = new PdfExport();
    // Here the 50f is the default value for the margins. If you are not
assigning any separate values like Margins.Left = 15 means 50f value will
be setting to all the Margin properties.
    PdfPageSettings pageSettings = new PdfPageSettings(50f);
    // Here you can assign the PdfPageSize enum value (ex: A3, A4)
    pageSettings.Size = PdfPageSize.A3;
    // Here Landscape value assigned to the orientation. You can set either
Landscape/Portrait for this Orientation property
    pageSettings.Orientation = PdfPageOrientation.Landscape;
    // Here assigned different values for the margins
    pageSettings.Margins.Left = 15;
    pageSettings.Margins.Right = 15;
    pageSettings.Margins.Top = 20;
    pageSettings.Margins.Bottom = 20;
    // Here assigned the Transition Direction as BottomToTop. You can also
set any of the Transition values to this property
    pageSettings.Transition.Direction = PdfTransitionDirection.BottomToTop;
    // Here assigned the Rotate Angle as RotateAngle180. You can also set any
of the Rotate values to this property
    pageSettings.Rotate = PdfPageRotateAngle.RotateAngle180;
    // Here passing the page settings object value into the export method.
    PdfDocument document = exp.Export(scheduleObject, scheduleAppointments,
    ExportTheme.FlatSaffron, Request.Form["locale"], pageSettings);
    document.Save("Schedule.pdf", HttpContext.ApplicationInstance.Response,
    HttpReadType.Save);
}
```

```
return RedirectToAction("PDFExportController");
}
```

### Header/Footer Settings

You can also set the header and footer options to the PDF page before it is being exported, by passing the `PDFDocumentTemplate` object values as mentioned in the following code example.

### C#

```
public ActionResult PDFExport(string scheduleModel)
{
    JavaScriptSerializer serializer = new JavaScriptSerializer();
    SchedulePDFExport convert = new SchedulePDFExport();
    ScheduleProperties scheduleObject =
        convert.ScheduleSerializeModel(scheduleModel); // Here calling the public
method to convert the model values to ScheduleProperties type from string
type
    IEnumerable scheduleAppointments =
        (IEnumerable)serializer.Deserialize(Request.Form["ScheduleProcesedApps"],
        typeof(IEnumerable)); // Here deserialize the appointments to enumerable
object
    PdfExport exp = new PdfExport();
    PdfDocument document = new PdfDocument();
    PdfPage pdfPage = document.Pages.Add();
    //Create a header and draw the string.
    // Here the bounds value is used to store the position/axis value, where
the header and footer content to be displayed
    RectangleF bounds = new RectangleF(0, 0,
        document.Pages[0].GetClientSize().Width, 50);
    // Creating object for the PdfPageTemplateElement
    PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
    // Here setting the font style and color to display the header/footer
content
    PdfSolidBrush brush = new PdfSolidBrush(Color.Gray);
    PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 6,
        PdfFontStyle.Bold);
    PdfStringFormat format = new PdfStringFormat();
    format.Alignment = PdfTextAlignment.Center;
    format.LineAlignment = PdfVerticalAlignment.Middle;
    header.Graphics.DrawString("Schedule", font, brush, new RectangleF(0, 18,
        header.Width, header.Height), format);
    //Create a Page template that can be used as footer (here creating
template to display the page number).
    PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds);
    //Create page number field.
    PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
    //Create page count field.
    PdfPageCountField count = new PdfPageCountField(font, brush);
    //Add the fields in composite fields.
    PdfCompositeField compositeField = new PdfCompositeField(font, brush,
        "Page {0} of {1}", pageNumber, count);
    compositeField.Bounds = footer.Bounds;
    //Draw the composite field in footer.
    compositeField.Draw(footer.Graphics, new PointF(470, 40));
    PdfDocumentTemplate headerFooterTemplate = new PdfDocumentTemplate();
```

```
//Here assigning the header template value to the PdfDocumentTemplate
Object.
headerFooterTemplate.Top = header;
//Here assigning the footer template value to the PdfDocumentTemplate
Object.
headerFooterTemplate.Bottom = footer;
//Here passing the PdfDocumentTemplate Object value into the export
method.
document = exp.Export(scheduleObject, scheduleAppointments,
ExportTheme.FlatSaffron, Request.Form["locale"], headerFooterTemplate);
document.Save("Schedule.pdf", HttpContext.ApplicationInstance.Response,
HttpReadType.Save);
return RedirectToAction("PDFExportController");
}
```

**Note:** The header and footer values can be passed to the PDF file only through the template option. In the above mentioned code example, the template has been written with the text “Schedule” to display it as the header text and the page number (ex: Page 1 of 2) has been displayed in the footer part.

It is also possible to pass both the `PageSettings` and header/footer template objects in the export method altogether to apply the page settings as well as the header and footer options to the PDF file which can be referred from the following code example,

### C#

```
public ActionResult PDFExport(string scheduleModel)
{
//Here add the code example of both the page settings and header/footer
values settings.
//Then pass both PdfPageSettings and PdfDocumentTemplate object into the
export method.
document = exp.Export(scheduleObject, scheduleAppointments,
ExportTheme.FlatSaffron, Request.Form["locale"], pageSettings,
headerFooterTemplate);
document.Save("Schedule.pdf", HttpContext.ApplicationInstance.Response,
HttpReadType.Save);
return RedirectToAction("PDFExportController");
}
```

### Excel Export

Scheduler supports exporting all appointments in Excel format, for which the same [saveAsExcel](#) method can be used with passing type value to its parameter list. To achieve this, keep an individual button to export and when it is clicked, the appointments can be allowed to export as Excel.

In [saveAsExcel](#) method, we can pass three arguments as follows:

- Action name (to be called in the server-side)
- Server Event (Optional)
- Appointment export type (including occurrence appointments or excluding occurrence appointments)

The appointment has been exported based on the class fields. We should create the class with the fields to be exported and based on that class, object can be deserialized.

The following code example depicts the way to export the appointments in Excel format.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<!-- Button div for Export Option-->
<button id="btnExport">Export</button>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var button = new ej.Button($("#btnExport"), {
width: "80px",
height: "30px",
click: "onExportClick"
});
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
height: "525px",
currentDate: new Date(2014, 4, 5),
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
});
});
}
```

### JAVASCRIPT

```
function onExportClick(e) {
var obj = $("#Schedule1").data("ejSchedule");
obj.saveAsExcel("http://js.syncfusion.com/ejservices/api/Schedule/XlsExport", null, false);
e.cancel = true;
}
```

The server-side action **ExportAsExcel** contains the following code example to export the Scheduler.

### C#

```
public ActionResult ExportAsXSL()
{
List<AppointmentData> scheduleAppointments =
(List<AppointmentData>) JsonConvert.DeserializeObject(Request.Form["ScheduleAppointment"], typeof(List<AppointmentData>));
ExcelExport xlExport = new ExcelExport();
```



```

return xlExport.Export(scheduleAppointments, ExcelVersion.Excel2013);
}
public class AppointmentData
{
    public int Id { get; set; }
    public string Subject { get; set; }
    public string StartTime { get; set; }
    public string EndTime { get; set; }
    public bool AllDay { get; set; }
    public bool Recurrence { get; set; }
    public string RecurrenceRule { get; set; }
}

```

In Export method, the appointments are mandatory to pass as argument to export in excel format. In addition, we can pass filename and excel version as optional parameter to export file with specified filename and version.

### Print

Two types of Print options are available – either to print the entire Scheduler layout including all the appointments in it or else to print any particular appointment alone. The public method [print](#) is used to print the entire Scheduler.

#### Print the Scheduler

The following code example shows the way to print the entire Scheduler, by keeping an extra button in a page – on which clicking the button will print the entire Scheduler.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<!--Button div for Print Option-->
<button id="printBtn">Print</button>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
    $(function () {
        var button = new ej.Button($("#printBtn"), {
            width: "70px",
            height: "30px",
            click: "onClick"
        });
        var sample = new ej.Schedule($("#Schedule1"), {
            currentDate: new Date(2015, 11, 2),
            appointmentSettings: {
                dataSource: [{
                    Id: 100,
                    Subject: "Wild Discovery",
                    StartTime: new Date(2015, 11, 2, 9, 00),
                    EndTime: new Date(2015, 11, 2, 10, 30),
                    Location: "CHINA"
                }]
            }
        });
    });
}

```

```
});
});
}
```

## JAVASCRIPT

```
function onClick(args) {
var obj = $("#Schedule1").data("ejSchedule");
obj.print();
}
```

### Printing Specific Appointments

To print a particular appointment, either the default context menu **print** option can be used or else the [print](#) method can be used by passing the id of the appointment to be printed as its argument.

### Using Context Menu

Here, the print action is handled internally by default, so that when an appointment is right clicked – choosing print option from the context menu that pops-out will automatically print that particular appointment.

**Note:** To handle this functionality, the context menu settings needs to be enabled with print option added to the appointment items.

The following code example depicts the way to print a particular appointment.

## HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"> </div>
```

## JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
contextMenuSettings: {
enable: true,
menuItems: {
appointment: [
{ id: "open", text: "Open Appointment" },
{ id: "delete", text: "Delete Appointment" },
{ id: "print", text: "Print Appointment" }
]
}
},
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA"
}]
}
```

```

    }]
  }
  });
});
}

```

### Using print() method

The public method `print` needs to be used here by passing the appointment object as its argument, that needs to be printed.

For example, here the below code example depicts the way to print the particular appointment programmatically by calling the `print` function within the [appointmentClick](#) event, which triggers whenever the user clicks on an appointment.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"> </div>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
  $(function () {
    var sample = new ej.Schedule($("#Schedule1"), {
      currentDate: new Date(2015, 11, 2),
      appointmentSettings: {
        dataSource: [{
          Id: 100,
          Subject: "Wild Discovery",
          StartTime: new Date(2015, 11, 2, 9, 00),
          EndTime: new Date(2015, 11, 2, 10, 30),
          Location: "CHINA"
        }]
      },
      appointmentClick: "onAppointmentClick"
    });
  });
}

```

### JAVASCRIPT

```

function onAppointmentClick(args) {
  var schObj = $("#Schedule1").data("ejSchedule");
  schObj.print(args.appointment);
}

```

### Import Appointments

To import appointments into the Scheduler, server-side method `renderingImportAppointments` needs to be used, which returns the list of appointments retrieved from the specified file path.

Refer the following code example to import the appointments into Scheduler.

**HTML**

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
<!--Button div for Print Option-->
<button id="importBtn">Import</button>

```

**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var button = new ej.Button($("#importBtn"), {
width: "70px",
height: "30px",
click: "ScheduleImport"
});
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
height: "525px",
currentDate: new Date(2015, 11, 2),
appointmentSettings: {
id: "Id",
subject: "subject",
startTime: "startTime",
endTime: "endTime",
allDay: "allDay",
recurrence: "recurrence",
recurrenceRule: "recurrenceRule"
}
});
});
});
}

```

**JAVASCRIPT**

```

function ScheduleImport() {
var dataManger = ej.DataManager({ url: '@Url.Action("ScheduleImportData",
"Schedule")', crossDomain: true });
$("#Schedule1").ejSchedule({
appointmentSettings: {
dataSource: dataManger
}
});
}

```

The server-side action **ScheduleImportData** contains the following code example to import the Scheduler appointments.

**C#**

```

public JsonResult ScheduleImportData() {
var destinationPath = @"FilePath\iCalender.ics";
ScheduleImport importApps = new ScheduleImport();

```

```

var app = importApps.renderingImportAppointments(destinationPath);
int intMax = app.Max(a => a.Id);
List<ScheduleAppointmentsObjData> result = new
List<ScheduleAppointmentsObjData>();
for (var i = 0; i < app.Count; i++) {
app[i].Id = intMax + 1;
ScheduleAppointmentsObjData row = new
ScheduleAppointmentsObjData(app[i].Id, app[i].Subject, app[i].Location,
app[i].StartTime, app[i].EndTime, app[i].Description, null, null,
app[i].Recurrence, null, null, app[i].AppointmentCategorize, null,
app[i].AllDay, null, null, app[i].RecurrenceRules, null, null);
result.Add(row);
intMax = app[i].Id;
}
return Json(result, JsonRequestBehavior.AllowGet);
}

```

## Miscellaneous

### Time Indicator

The current system time can be depicted in a scheduler with an indication of a piece of text displaying the time over a horizontal line across today's date (Vertical Scheduler mode). In Horizontal mode, the time indicator is displayed as a small vertical line within the header time cells (depicting current time).

It is enabled by default in Scheduler and to hide it, [showCurrentTimeIndicator](#) property needs to be set as **false** as shown below.

### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
showCurrentTimeIndicator: false,
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30)
}]
}
});
});
}

```

**Note:** When Scheduler rendered with multiple resources, multiple time indicators are displayed in Vertical mode – whereas single indicator displays for horizontal mode.

### Show/Hide All-Day Row

The All-day row cell is a single row region displayed at the top of the work cells area to hold the all-day appointments together. The Scheduler displays it by default and if it needs to be hidden, the [showAllDayRow](#) property can be set to **false** as shown below.

#### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
showAllDayRow: false,
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30)
}]
}
});
});
}
```

**Note:** The All-day row expands vertically, whenever more number of appointments are populated in one or more days.

### Show/Hide Header bar

The header bar is the top most region of the Scheduler which includes the date navigation icons and view navigation options – and also shown on the Scheduler by default. To hide the header bar, set the [showHeaderBar](#) property to **false** as shown below.

#### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
showHeaderBar: false,
appointmentSettings: {
dataSource: [{
```

```

Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30)
}]
}
});
});
}

```

### Show/Hide TimeScale

The TimeScale is the left most region of the Scheduler in vertical mode and the same is displayed at the top position in horizontal mode. It depicts the time interval either in a 12 or 24 hour mode. By default, the timeScale is displayed in the Scheduler and in order to hide it – set the [showTimeScale](#) option to **false** as shown below.

#### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

#### JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
showTimeScale: false,
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30)
}]
}
});
});
}

```

### Show/Hide Location Field

The Location field in the default appointment window can be displayed or hidden from it using the [showLocationField](#) property. By default, this property is set to false. To display the location option in the appointment window, then set **showLocationField** to **true** and also bind the appropriate field to the location property within the appointmentSettings as shown below.

#### HTML

```

<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
showLocationField: true,
appointmentSettings: {
location: "EventLocation",
dataSource: [{
Id: 100,
Subject: "Research on Sky Miracles",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
EventLocation: "RDU"
}]
}
});
});
}

```

**Recurrence Editor**

The **Recurrence Editor** includes the entire recurrence related information in a separate portable manner which can be either utilized as a separate widget or else can be embed within the appointment window of Scheduler to enable recurrence options within it. The recurrence rule can be easily generated based on the frequency selected. The customizations like changing the labels of the recurrence editor is also possible to achieve through its properties. The frequencies available are Never, Daily, Weekly, Monthly, Yearly and Every weekday.

**Getting Started**

To render the Recurrence Editor control as a separate widget, the following list of external dependencies are needed,

- [jQuery](#) - 1.7.1 and later versions
- [jsRender](#) - to render the templates
- [jQuery.easing](#) - to support animation effects in the components

The other required internal dependencies are tabulated below,

| File                       | Description/Usage  |
|----------------------------|--|
| ej.core.min.js             | Must be referred always first before using all the JS controls.        |
| ej.globalize.min.js        | Must be referred to localize any of the JS control's text and content. |
| ej.recurrenceeditor.min.js | Schedule core file   |



|                        |   |
|------------------------|---|
| ej.scroller.min.js     | These files are referred for proper working of the sub-controls used within RecurrenceEditor. |
| ej.datepicker.min.js   |   |
| ej.checkbox.min.js     |   |
| ej.editor.min.js       |   |
| ej.dropdownlist.min.js |   |
| ej.radiobutton.min.js  |   |

**Note:** Recurrence Editor uses one or more sub-controls, therefore refer the `ej.web.all.min.js` (which encapsulates all the `ej` controls and frameworks in a single file) in the application instead of referring all the above specified internal dependencies.

To get the real appearance of the Recurrence Editor, the dependent CSS file `ej.web.all.min.css` (which includes styles of all the widgets) should also needs to be referred.

### Control Initialization

Create a div element within the body section of the HTML document, where the Recurrence Editor needs to be rendered.

### HTML

```
<body>
<div id="RecurrenceEditor"></div>
</body>
```

Initialize the Recurrence Editor control, by adding the following script code to the body section of the HTML document.

### HTML

```
<!-- div for RecurrenceEditor creation -->
<div id="RecurrenceEditor"></div>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module RecurrenceComponent {
  $(function () {
    var sample = new ej.RecurrenceEditor($("#RecurrenceEditor"));
  });
}
```

### Generating Recurrence Rule

The Recurrence Editor can be used to generate the recurrence rule as a string, based on the repeat options selected.

The following code example depicts the way to generate the recurrence rule.

### HTML

```
<!--Container for ejRecurrenceEditor widget-->
<div id="RecurrenceEditor"></div>
<button type="donerecur" id="donerecur1" class='recurbutton'
style="float:right;margin-right:20px;margin-bottom:10px;">Recurrence
String</button>
```

### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module RecurrenceComponent {
$(function () {
var button = new ej.Button($("#donerecur1"), {
width: '155px',
height: '35px',
showRoundedCorner: true,
click: "closeRecurrence"
});
var sample = new ej.RecurrenceEditor($("#RecurrenceEditor"), {
selectedRecurrenceType: 0,
create:"onCreate"
});
});
});
}
```

### JAVASCRIPT

```
function onCreate() {
this.element.find("#recurrencetype_wrapper").css("width", "33%");
$("#RecurrenceEditor").after($("#donerecur1"));
}
function closeRecurrence() {
var obj = $("#RecurrenceEditor").data("ejRecurrenceEditor");
alert(obj.getRecurrenceRule());
}
```

## How To

### Validate the Custom Appointment Window Fields

The client-side validation of the fields present within the custom appointment window can be handled before submitting it, by adding appropriate validation classes to each field.

Refer the steps [here](#) and create a sample for Custom Appointment window, before proceeding with the following validations.

In the custom appointment window sample, create an additional CSS class **validation** as mentioned below to add it to the appropriate fields, if the validation of such fields fails.

### HTML

```
<style>
.validation {
```

```
border-color: red;
}
</style>
```

The sample contains the fields like Subject, Description, StartTime and EndTime which needs to be validated at client-side. To do so, add the required validation code within the script section as follows.

### JAVASCRIPT

```
// To Validate the Subject field.
$("#subject").focusout(function() {
if ($.trim($("#subject").val()) == "") {
$("#subject").addClass("validation");
return false;
}
});
// To Validate the Description field.
$("#customDescription").focusout(function() {
if ($.trim($("#customDescription").val()) == "") {
$("#customDescription").addClass("validation");
return false;
}
});
// To Validate the Time duration of the appointments
$("#EndTime").focusout(function() {
if (new Date($("#EndTime").val()).getDate() >= new
Date($("#StartTime").val()).getDate()) {
if (new Date($("#StartTime").val()).getTime() >= new
Date($("#EndTime").val()).getTime())
alert("EndTime value is lesser than the StartTime value");
}
});
```

Now, after adding the above validations – whenever the fields within the custom appointment window are skipped without filling any information, it will be notified to the users appropriately.

### Highlight Different Work Hours for Each Resources

By default, the work hours of the Scheduler is highlighted based on the start and end values provided within the [workHours](#) object. It remains same for all the resources, when the Scheduler is rendered with multiple resources. To customize this behavior so as to highlight different work hours range for each of the resources, the following workaround can be utilized by making use of the Scheduler events **create** and [actionComplete](#).

Initially, set the **highlight** as false for the **workHours**, so as to disable the highlighting of default work hour range.

### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

### JAVASCRIPT

**JAVASCRIPT**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
showCurrentTimeIndicator: false,
workHours: {
highlight: false
},
group: {
resources: ["Owners"]
},
resources: [{
field: "ownerId",
title: "Owner",
name: "Owners",
resourceSettings: {
dataSource: [
{ text: "Nancy", id: 1, color: "#f8a398", on: 10, off: 18 },
{ text: "Steven", id: 3, color: "#56ca85", on: 6, off: 10 },
{ text: "Michael", id: 5, color: "#51a0ed", on: 11, off: 15 }
],
text: "text",
id: "id",
color: "color",
start: "on",
end: "off"
}
}],
appointmentSettings: {
resourceFields: "ownerId",
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
ownerId: 3,
Location: "CHINA"
}]
}
});
});
}

```

{% endhighlight %}

**Display Scheduler with Appointments Filtered by Subject**

It is possible to display the Scheduler with appointments, which is filtered based on the Subject. For example, if we keep a text box and start typing a character – the list of appointments whose subject matches the typed character alone will be shown on the Scheduler. The following code example depicts the way to achieve this.

**HTML**

```

<!--textbox for entering search character-->
<input id='txtSearch' type='text' onkeyup='searchKeyUp()' />
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

## JAVASCRIPT

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
// Appointment data to be bound to the Scheduler
window.Default = [
{
Id: 101,
Subject: "Bering Sea Gold",
StartTime: new Date(2015, 11, 5, 10, 00),
EndTime: new Date(2015, 11, 5, 11, 00),
Description: "",
AllDay: false,
Recurrence: false,
Categorize: "1,3"
}, {
Id: 102,
Subject: "Bering Sea Gold",
StartTime: new Date(2015, 11, 2, 16, 00),
EndTime: new Date(2015, 11, 2, 17, 30),
Description: "",
AllDay: false,
Recurrence: false,
Categorize: "2,5"
}, {
Id: 104,
Subject: "What Happened Next?",
StartTime: new Date(2015, 11, 5, 12, 30),
EndTime: new Date(2015, 11, 5, 15, 00),
Description: "",
AllDay: false,
Recurrence: false,
Categorize: "4,1"
}, {
Id: 105,
Subject: "Daily Planet",
StartTime: new Date(2015, 11, 3, 01, 00),
EndTime: new Date(2015, 11, 3, 02, 00),
Description: "",
AllDay: false,
Recurrence: false,
Categorize: "1,3,6"
}, {
Id: 107,
Subject: "How It's Made",
StartTime: new Date(2015, 11, 1, 06, 00),
EndTime: new Date(2015, 11, 1, 07, 30),
Description: "",
AllDay: false,

```

```

Recurrence: true,
RecurrenceRule: "FREQ=WEEKLY;BYDAY=MO,TU;INTERVAL=1;COUNT=15",
Categorize: "2,3,6"
}, {
Id: 108,
Subject: "Deadest Catch",
StartTime: new Date(2015, 11, 3, 16, 00),
EndTime: new Date(2015, 11, 3, 17, 00),
Description: "",
AllDay: false,
Recurrence: false,
Categorize: "2,4,6,1"
}, {
Id: 109,
Subject: "MayDay",
StartTime: new Date(2015, 3, 30, 06, 30),
EndTime: new Date(2015, 3, 30, 07, 30),
Description: "",
AllDay: false,
Recurrence: false,
Categorize: "5,3"
}, {
Id: 115,
Subject: "Cash Cab",
StartTime: new Date(2015, 3, 30, 15, 00),
EndTime: new Date(2015, 3, 30, 16, 30),
Description: "",
AllDay: false,
Recurrence: true,
RecurrenceRule: "FREQ=DAILY;INTERVAL=1;COUNT=5",
Categorize: "1,3"
}
];
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 5),
showCurrentTimeIndicator: false,
appointmentSettings: {
dataSource: dManager
}
});
});
}

```

## JAVASCRIPT

```

// This function executes when a character is entered in the textbox
function searchKeyUp() {
var searchString = $("#txtSearch").val();
var schObj = $("#Schedule1").data("ejSchedule");
var result = schObj.searchAppointments(searchString);
schObj.option("appointmentSettings", { dataSource: [] });
if (!ej.isNullOrUndefined(result) && result.length != 0 && searchString != "")
schObj.option("appointmentSettings", { dataSource: result });
else
schObj.option("appointmentSettings", { dataSource: window.Default });
}

```

```
}

```

### Customize the Default Appointment Window

Apart from the custom appointment window, it is possible to customize the default appointment window by adding/removing the required number of fields into it. This can be achieved through the **appointmentWindowOpen** event of the scheduler.

The following code example depicts the way to achieve the customization of default appointment window.

#### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>

```

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var sample = new ej.Schedule($("#Schedule1"), {
currentDate: new Date(2015, 11, 2),
showCurrentTimeIndicator: false,
appointmentWindowOpen: "onAppointmentOpen",
appointmentSettings: {
dataSource: [{
Id: 100,
Subject: "Wild Discovery",
StartTime: new Date(2015, 11, 2, 9, 00),
EndTime: new Date(2015, 11, 2, 10, 30),
Location: "CHINA",
AppointmentType: "Tentative",
Status: "90%"
}]
}
});
});
}

```

#### JAVASCRIPT

```
// This function executes before the appointment window gets opened.
function onAppointmentOpen(args) {
// to add custom element in default appointment window
if (this._appointmentAddWindow.find(".custom-fields").length == 0) {
var customDesign = "<tr class='custom-fields'><td class='e-textlabel'>Event Type</td><td><input class='app-type' type='text' /></td><td class='e-textlabel'>Event Status </td><td><input class='status' type='text' /></td></tr>";
$(customDesign).insertAfter(this._appointmentAddWindow.find("." + this._id + "appointmentArrow"));
}
if (!ej.isNullOrUndefined(args.appointment)) {

```

```
// if double clicked on the appointments, retrieve the custom field
values from the appointment object and fills it in the appropriate
fields.
this._appointmentAddWindow.find(".app-
type").val(args.appointment.AppointmentType);
this._appointmentAddWindow.find(".status").val(args.appointment.Status);
} else {
// if double clicked on the cells, clears the field values.
this._appointmentAddWindow.find(".app-type").val("");
this._appointmentAddWindow.find(".status").val("");
}
}
```

### Synchronize the Schedule with Outlook

Schedule appointments can be synchronized with Outlook and vice versa, by making use of the Microsoft Outlook 12/15 Object library.

The following code example depicts the way to synchronize the Schedule with Outlook.

#### HTML

```
<!--Container for ejScheduler widget-->
<div id="Schedule1"></div>
```

#### JAVASCRIPT

```
/// <reference path="../tsfiles/jquery.d.ts" />
/// <reference path="../tsfiles/ej.web.all.d.ts" />
module ScheduleComponent {
$(function () {
var dataManager = ej.DataManager({
url: '@Url.Action("GetApp", "Home")',
crudUrl: '@Url.Action("Batch", "Home")',
adaptor: new ej.UrlAdaptor(),
crossDomain: true
});
var sample = new ej.Schedule($("#Schedule1"), {
width: "100%",
height: "525px",
currentDate: new Date(2015, 5, 15),
appointmentSettings: {
dataSource: dataManager,
id: "Id",
subject: "Subject",
startTime: "StartTime",
endTime: "EndTime",
startTimeZone: "StartTimeZone",
endTimeZone: "EndTimeZone",
allDay: "AllDay",
recurrence: "Recurrence",
recurrenceRule: "RecurrenceRule"
}
});
});
}
```



Schedule control is not directly compatible with the Outlook calendar object, as it returns the COM object. Therefore, in order to bind the schedule control with those data retrieved from outlook, then it is necessary to convert the COM object into the schedule acceptable object format. To do so add the following code example in your code behind.

### C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using Microsoft.Office.Interop.Outlook; // required Microsoft Assembly
using System.Web.Script.Serialization;
using System.Web.UI;
using System.Web.UI.WebControls;
using ScheduleCRUDJS.Models;
using System.Collections;
namespace ScheduleCRUDJS.Controllers
{
    public class HomeController : Controller
    {
        ScheduleDataDataContext db = new ScheduleDataDataContext();
        public ActionResult Index()
        {
            return View();
        }
        [HttpPost]
        public JsonResult GetApp() // function to display the outlook
            appointments in Schedule initially
        {
            IEnumerable data = GetData();
            return Json(data, JsonRequestBehavior.AllowGet);
        }
        public List<MultipleResource> GetData() // function to return the outlook
            appointments
        {
            Microsoft.Office.Interop.Outlook.Application oApp = new
            Microsoft.Office.Interop.Outlook.Application();
            Microsoft.Office.Interop.Outlook.NameSpace mapNamespace =
            oApp.GetNamespace("MAPI");
            Microsoft.Office.Interop.Outlook.MAPIFolder CalendarFolder =
            mapNamespace.GetDefaultFolder(Microsoft.Office.Interop.Outlook.OlDefaultF
            olders.olFolderCalendar);
            Microsoft.Office.Interop.Outlook.Items outlookCalendarItems =
            CalendarFolder.Items;
            List<Microsoft.Office.Interop.Outlook.AppointmentItem> appointmentList =
            new List<Microsoft.Office.Interop.Outlook.AppointmentItem>();
            foreach (Microsoft.Office.Interop.Outlook.AppointmentItem item in
            outlookCalendarItems)
            {
```

```

appointmentList.Add(item);
}
List<MultipleResource> newApp = new List<MultipleResource>();
// converting COM object into IEnumerable object
for (var i = 0; i < appointmentList.Count; i++)
{
    MultipleResource app = new MultipleResource();
    app.Id = i;
    app.Subject = appointmentList[i].Subject;
    app.AllDay = appointmentList[i].AllDayEvent;
    app.StartTime = Convert.ToDateTime(appointmentList[i].Start.ToString());
    string endTime = appointmentList[i].End.ToString();
    DateTime appEndDate = Convert.ToDateTime(endTime);
    if (endTime.Contains("12:00:00 AM") && app.AllDay == true)
        app.EndTime = appEndDate.AddMinutes(-1);
    else
        app.EndTime = appEndDate;
    app.Recurrence = false;
    app.RecurrenceRule = null;
    newApp.Add(app);
}
return newApp;
}

public JsonResult Batch(EditParams param)
{
    if (param.action == "insert" || (param.action == "batch" && param.added
        != null)) // this block of code will execute while inserting the
        appointments
    {
        var value = param.action == "insert" ? param.value : param.added[0];
        int intMax = db.MultipleResources.ToList().Count > 0 ?
            db.MultipleResources.ToList().Max(p => p.Id) : 1;
        DateTime startTime = Convert.ToDateTime(value.StartTime);
        DateTime endTime = Convert.ToDateTime(value.EndTime);
        MultipleResource appoint = new MultipleResource()
        {
            Id = intMax + 1,
            StartTime = startTime,
            EndTime = endTime,
            StartTimeZone = value.StartTimeZone,
            EndTimeZone = value.EndTimeZone,
            Subject = value.Subject,
            OwnerId = value.OwnerId,
            AllDay = value.AllDay,
            Recurrence = value.Recurrence,
            RecurrenceRule = value.RecurrenceRule
        };
        db.MultipleResources.InsertOnSubmit(appoint);
        db.SubmitChanges();
        Microsoft.Office.Interop.Outlook.Application outlookApp = new
            Microsoft.Office.Interop.Outlook.Application();
        Microsoft.Office.Interop.Outlook.AppointmentItem newAppointment = null;
        newAppointment =
            (Microsoft.Office.Interop.Outlook.AppointmentItem) outlookApp.CreateItem(M
                icrosoft.Office.Interop.Outlook.OlItemType.olAppointmentItem);
        newAppointment.Start =
            Convert.ToDateTime(value.StartTime).ToUniversalTime();
    }
}

```

```
newAppointment.End = Convert.ToDateTime(value.EndTime).ToUniversalTime();
newAppointment.Subject = value.Subject.ToString();
newAppointment.Save();
}
IEnumerable data = new
ScheduleDataDataContext().MultipleResources.Take(200);
return Json(data, JsonRequestBehavior.AllowGet);
}
}
}
```

**Note:** In order to achieve the above scenario, need to refer the Microsoft assembly in your application (Microsoft Outlook 12/15 Object library [Microsoft.Office.Interop.Outlook] and refer it in the controller page as shown above).

## Scroller

### Overview

The **Scroller** control has a sliding document whose position corresponds to a value. The document has text, HTML content or images. You can also customize the Scroller control by resizing the scrolling bar and changing the theme.

### Key Features

- **Height and Width** - Set the height and width of the scroll panel.
- **Button Size** - Customize the width and height of the buttons (UP, DOWN, RIGHT and LEFT).
- **Scroller Size** - Customize the scrollbar width and height.
- **Step Increment** - Set the number of pixels to be moved on pressing the ARROW key.
- **RTL Support** - Sets the alignment of the horizontal Scroller to the right, the reading order to right-to-left and the layout of the control to flow from right to left.
- **Theme** - Essential JavaScript controls consists of 12 built in themes ( 6 – flat and 6 – gradient effects), and also supports custom skins to set user-defined themes.

### Create a simple Scroller in TypeScript

You can create a **TypeScript** application with the help of the given

<https://help.syncfusion.com/js/typescript>.

Create an **HTML** page and add the scripts references in the order, mentioned in the above link Create the **Scroller** control as follows.

### HTML

```
<body>
<div id="scrollcontent">
<div>
<div class="sampleContent">
<h3 style="font-size: 20px;">MVC</h3>
<div>
<p>
Model-view-controller (MVC) is a software architecture pattern which
separates the
representation of information from the user's interaction with it.
```

The model consists of application data, business rules, logic, and functions. A view can be any output representation of data, such as a chart or a diagram. Multiple views of the same data are possible, such as a bar chart for management and a tabular view for accountants.

The controller mediates input, converting it to commands for the model or view. The central ideas behind MVC are code reusability and in addition to dividing the application into three kinds of components, the MVC design defines the interactions between them.

</p>

<ul>

<li>

<b>A controller </b>can send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document).

It can also send commands to the model to update the model's state (e.g., editing a document).

</li>

<li>

<b>A model</b> notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands.

A passive implementation of MVC omits these notifications, because the application does not require them or the software platform does not support them.

</li>

<li>

<b>A view</b> requests from the model the information that it needs to generate an output representation to the user.

</li>

</ul>

</div>

</div>

</div>

</div>

</body>

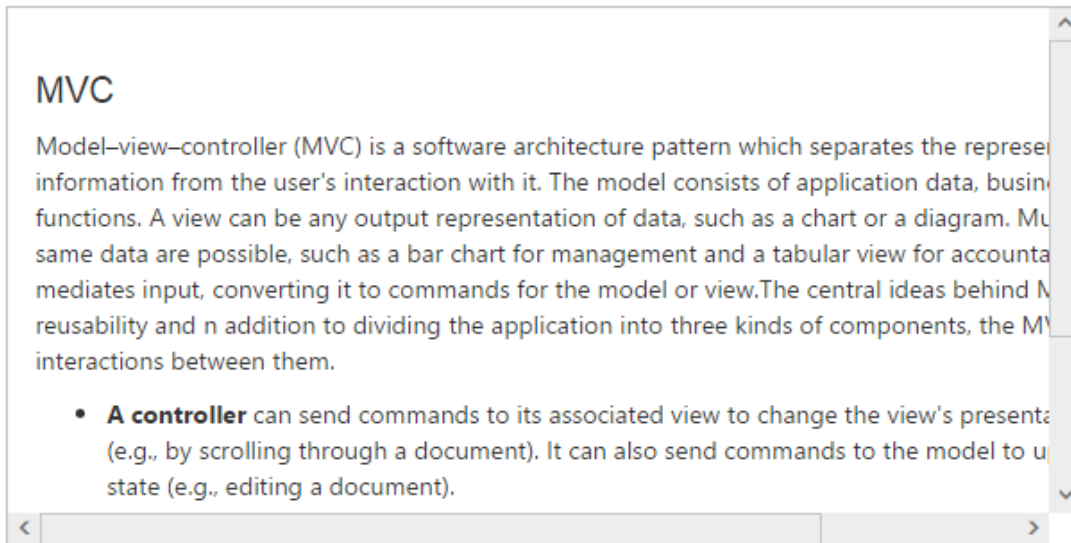
## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ScrollerComponent {
  $(function () {
    var scrollerSample = new ej.Scroller($("#scrollcontent"), {
      height: "300px",
      width: "100%"
    });
  });
}

```

You can execute the above code example to display the **Scroller** control.



## Scroller Styles

The **Essential JavaScript Scroller** control allows you to customize the look and function of scrollbars. You can vary it significantly by setting the scrollbar button size, scrollbar position, height and width of the **Scroller** control. This section describes you the custom styles to be used when creating **Scroller**.

### Button Size

In Scroller control, it allows you to customize the scroll arrows width and height. In horizontal **scroller** the **buttonSize** customizes the top and down arrow and in vertical **scroller** the **buttonSize** customizes the left and right arrow.

### Scroller Size

The **scrollerSize** property is used to customize the scrollbar width and height. It is applicable for both horizontal and vertical scroller.

### Scroll Top

The **scrollerTop** property is used to move the **Scroller** content and scrollbars in top position with the specified value. It is used for only vertical scroller.

### Scroll Left

The **scrollerLeft** property is used to move the **Scroller** content and scrollbars in left position with the specified value. It is used for only horizontal scroller.

### Height

The height property is used to set the height for **Scroller** outer wrapper.

### Width

The width property is used to set the width for **Scroller** outer wrapper.

The following steps explains you on how to apply styles in **Scroller** control.

In the HTML page, add a <div> element to configure Scroller widget.

### HTML

```
<div class="content-container-fluid">
```

```
<div class="row">
<div class="cols-sample-area">
<div class="control">
<div id="scrollcontent">
<div>
<div class="sampleContent">
<h3 style="font-size: 20px;">MVC</h3>
<div>
<p>Model-view-controller (MVC) is a software architecture pattern which
separates the
representation of information from the user's interaction with it.
The model consists of application data, business rules, logic, and
functions. A view can be any
output representation of data, such as a chart or a diagram. Multiple
views of the same data
are possible, such as a bar chart for management and a tabular view for
accountants.
The controller mediates input, converting it to commands for the model or
view.The central
ideas behind MVC are code reusability and n addition to dividing the
application into three
kinds of components, the MVC design defines the interactions between
them.</p>
<ul>
<li>
<b>A controller </b>can send commands to its associated view to change
the view's presentation of the model (e.g., by scrolling through a
document).
It can also send commands to the model to update the model's state (e.g.,
editing a document).
</li>
<li>
<b>A model</b> notifies its associated views and controllers when there
has been a change in its state. This notification allows the views to
produce updated output, and the controllers to change the available set
of commands.
A passive implementation of MVC omits these notifications, because the
application does not require them or the software platform does not
support them.
</li>
<li>
<b>A view</b> requests from the model the information that it needs to
generate an output representation to the user.
</li>
</ul>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ScrollerComponent {
$(function () {
var scrollerSample = new ej.Scroller($("#scrollcontent"), {
height: 300,
width: 600,
scrollTop: 10,
scrollLeft: 20,
buttonSize: 20,
});
});
}

```

Configure the styles.

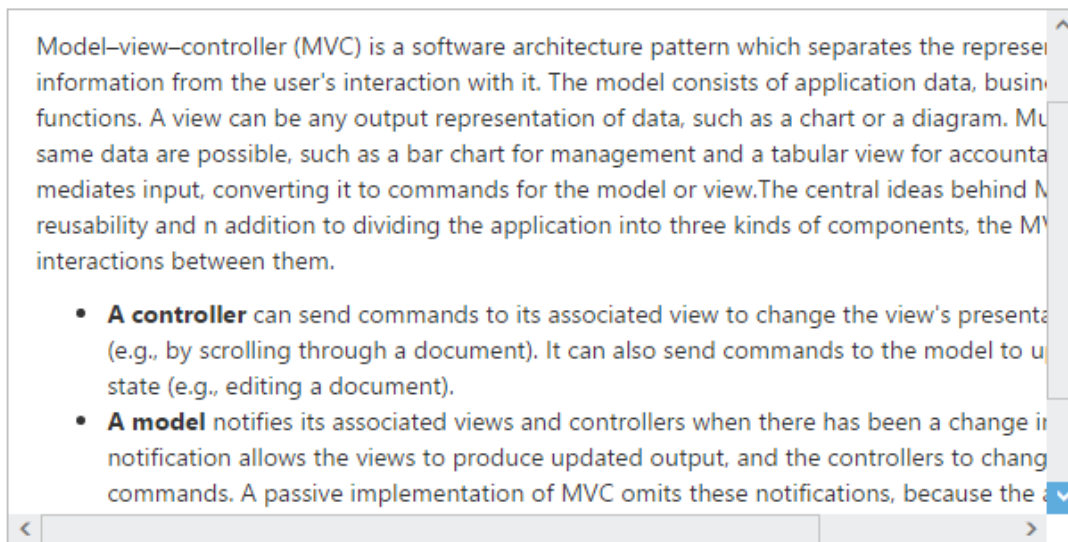
### CSS

```

<style type="text/css">
.control {
border: 1px solid #bbbcbb;
width: 600px;
margin: 0 auto;
height: 300px;
}
.sampleContent {
width: 700px;
padding: 15px;
}
</style>

```

The following screenshot displays the **Scroller** control with basic styles



## Thumb Scrolling

Normally the scrollbar position can be changed by dragging the scrollbar handle or clicking the arrows. The **Scroller** control allows you for panning or dragging the scroll content area to drag by dragging inside the scroll content. To achieve this in your **Scroller** control, enable the **enableTouchScroll** to true. By default the value for **enableTouchScroll** is true. When you want to prevent the panning or dragging the scroll content area, set **enableTouchScroll** as false.

The following steps explains you the configuration of **enableTouchScroll** property in **Scroller**.

In the HTML page, add a <div> element to configure Scroller widget.

### HTML

```
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<div class="control">
<div id="scrollcontent">
<div>
<div class="sampleContent">
<h3 style="font-size: 20px;">MVC</h3>
<div>
<p>Model-view-controller (MVC) is a software architecture pattern which
separates the
representation of information from the user's interaction with it.
The model consists of application data, business rules, logic, and
functions. A view can be any
output representation of data, such as a chart or a diagram. Multiple
views of the same data
are possible, such as a bar chart for management and a tabular view for
accountants.
The controller mediates input, converting it to commands for the model or
view.The central
ideas behind MVC are code reusability and n addition to dividing the
application into three
kinds of components, the MVC design defines the interactions between
them.</p>
<ul>
<li>
<b>A controller </b>can send commands to its associated view to change
the view's presentation of the model (e.g., by scrolling through a
document).
It can also send commands to the model to update the model's state (e.g.,
editing a document).
</li>
<li>
<b>A model</b> notifies its associated views and controllers when there
has been a change in its state. This notification allows the views to
produce updated output, and the controllers to change the available set
of commands.
A passive implementation of MVC omits these notifications, because the
application does not require them or the software platform does not
support them.
</li>
<li>
<b>A view</b> requests from the model the information that it needs to
generate an output representation to the user.
```



```

</li>
</ul>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>

```

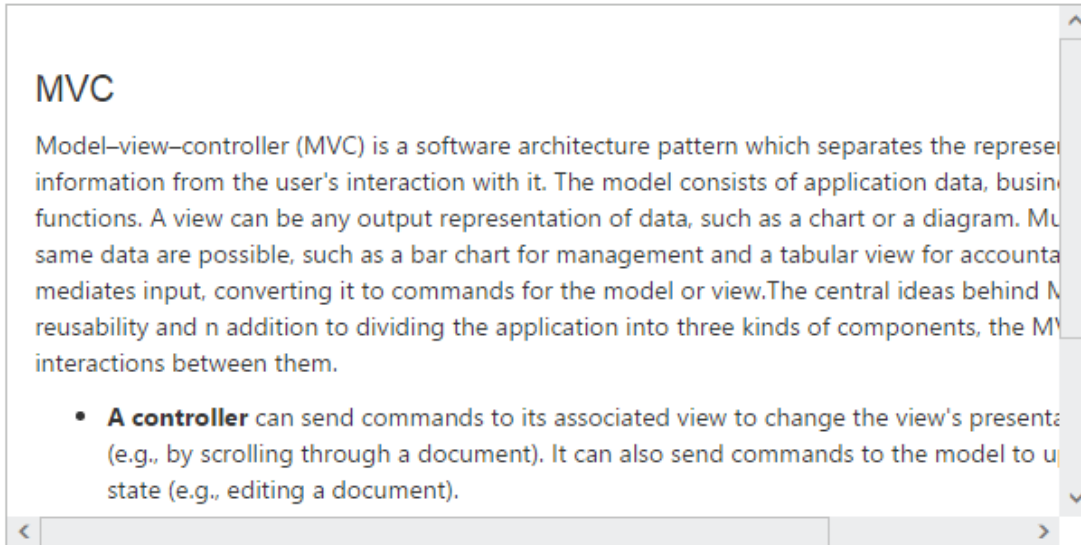
## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ScrollerComponent {
$(function () {
var scrollerSample = new ej.Scroller($("#scrollcontent"), {
height: 300,
width: 600,
enableTouchScroll: false
});
});
}

```

The following screenshot displays **Scroller** control with disabled touch support.



## Customizing the scroll Step

The **Scroller** control allows you to specify the scroll movement step in a pixel value, this step value is added to the scrollbar position when you press the arrow key. Based on position value, the scrollbar moves in its corresponding orientation. You can achieve this by using **scrollOneStepBy**.

The following steps explain you the configuration of **scrollOneStepBy** property in **Scroller**.

In the HTML page, add a <div> element to configure Scroller widget.

### HTML

```
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<div class="control">
<div id="scrollcontent">
<div>
<div class="sampleContent">
<h3 style="font-size: 20px;">MVC</h3>
<div>
<p>Model-view-controller (MVC) is a software architecture pattern which
separates the
representation of information from the user's interaction with it.
The model consists of application data, business rules, logic, and
functions. A view can be any
output representation of data, such as a chart or a diagram. Multiple
views of the same data
are possible, such as a bar chart for management and a tabular view for
accountants.
The controller mediates input, converting it to commands for the model or
view.The central
ideas behind MVC are code reusability and n addition to dividing the
application into three
kinds of components, the MVC design defines the interactions between
them.</p>
<ul>
<li>
<b>A controller </b>can send commands to its associated view to change
the view's presentation of the model (e.g., by scrolling through a
document).
It can also send commands to the model to update the model's state (e.g.,
editing a document).
</li>
<li>
<b>A model</b> notifies its associated views and controllers when there
has been a change in its state. This notification allows the views to
produce updated output, and the controllers to change the available set
of commands.
A passive implementation of MVC omits these notifications, because the
application does not require them or the software platform does not
support them.
</li>
<li>
<b>A view</b> requests from the model the information that it needs to
generate an output representation to the user.
</li>
</ul>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
```

```
</div>
```

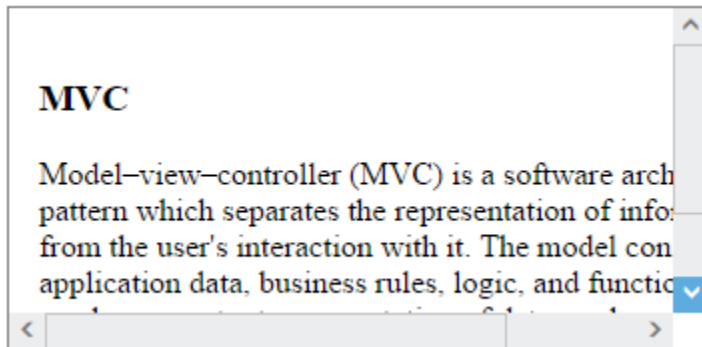
### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ScrollerComponent {
$(function () {
var scrollerSample = new ej.Scroller($("#scrollcontent"), {
height: 300,
width: 600,
scrollOneStepBy: 50
});
});
}
```

### CSS

```
<style type="text/css">
.control {
border: 1px solid #bbbcbb;
width: 600px;
margin: 0 auto;
height: 300px;
}
.sampleContent {
width: 700px;
padding: 15px;
}
</style>
```

The following screenshot displays the **Scroller** control with scroll step value.



### RTL

The **enableRTL** API provides right-to-left functionality and features for languages that work in a right-to-left for selecting, editing. Arabic and Hebrew are written from right to left. If you have a working style from right to left, you can use this feature in scroller. You can achieve this in your **Scroller** by using **enableRTL** property. Setting this property to true, the Scroller content text is displayed in the right to left format. The vertical scrollbar move to right to left side.

The following steps explains you the configuration of **enableRTL** property in **Scroller**.

In the HTML page, add a <div> element to configure Scroller widget.

### HTML

```
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<div class="control">
<div id="scrollcontent">
<div>
<div class="sampleContent">
<h3 style="font-size: 20px;">MVC</h3>
<div>
<p>Model-view-controller (MVC) is a software architecture pattern which
separates the
representation of information from the user's interaction with it.
The model consists of application data, business rules, logic, and
functions. A view can be any
output representation of data, such as a chart or a diagram. Multiple
views of the same data
are possible, such as a bar chart for management and a tabular view for
accountants.
The controller mediates input, converting it to commands for the model or
view.The central
ideas behind MVC are code reusability and n addition to dividing the
application into three
kinds of components, the MVC design defines the interactions between
them.</p>
<ul>
<li>
<b>A controller </b>can send commands to its associated view to change
the view's presentation of the model (e.g., by scrolling through a
document).
It can also send commands to the model to update the model's state (e.g.,
editing a document).
</li>
<li>
<b>A model</b> notifies its associated views and controllers when there
has been a change in its state. This notification allows the views to
produce updated output, and the controllers to change the available set
of commands.
A passive implementation of MVC omits these notifications, because the
application does not require them or the software platform does not
support them.
</li>
<li>
<b>A view</b> requests from the model the information that it needs to
generate an output representation to the user.
</li>
</ul>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
```

```
</div>
```

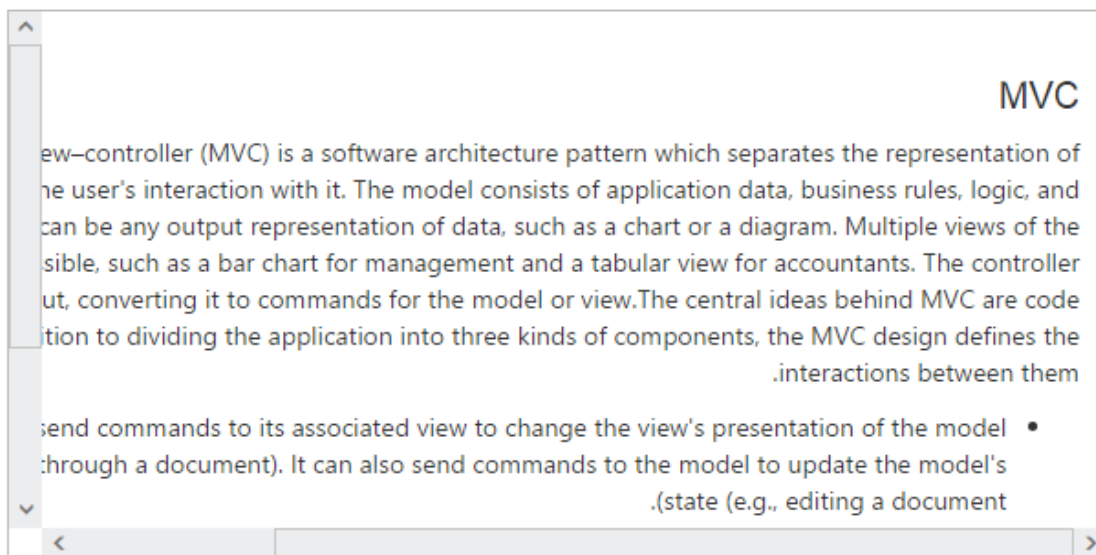
## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ScrollerComponent {
  $(function () {
    var scrollerSample = new ej.Scroller($("#scrollcontent"), {
      height: 300,
      width: 600,
      enableRTL: true
    });
  });
}
```

## CSS

```
<style type="text/css">
.control {
border: 1px solid #bbbcbb;
width: 600px;
margin: 0 auto;
height: 300px;
}
.sampleContent {
width: 700px;
padding: 15px;
}
</style>
```

The following screenshot displays the **Scroller** control in **RTL** direction.



## Signature

### Overview

The Essential Typescript Signature is a Plugin that is used to capture or drawing the smooth signatures. It captures user's signature as vector outlines of strokes. Using Signature we can customize the background, stroke width and stroke color and also convert captured signature to an image format. It is also provided with Undo, Redo & Clear options.

### Key Features

- **Height and Width** - Set the height and width of the signature canvas.
- **Stroke Color and Background Color** - Customize the stroke color and background color for the canvas
- **Background Image** - Sets the background image for the signature canvas.
- **Stroke Width** - Sets the stroke's minimum and maximum width for the drawing stroke.
- **Undo and Redo** – Undo and Redo the strokes in the signature, if wrongly drawn

### Getting Started

Using the following steps, you can create a **Typescript** Signature component. The basic rendering of **Typescript** Signature is achieved with default functionality.

#### Creating an Signature in Typescript

You can create a **Typescript** application with the help of the given [Typescript Getting Started Documentation](#).

Within an index.html file and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Typescript Application</title>
<link
href="http://cdn.syncfusion.com/{{site.releaseversion}}/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
<script
src="http://cdn.syncfusion.com/{{site.releaseversion}}/js/web/ej.web.all.
min.js" type="text/javascript"></script>
</head>
<body>
<!--Add Signature here-->
</body>
</html>
```

The Signature can be created from a HTML 'input' element with the HTML 'id' attribute and pre-defined options set to it. To create the Signature, you should call the ejSignature jQuery plug-in function.

#### HTML

```
<div style="width:300px;height:600px;">
<div id="signature" />
```

```
</div>
```

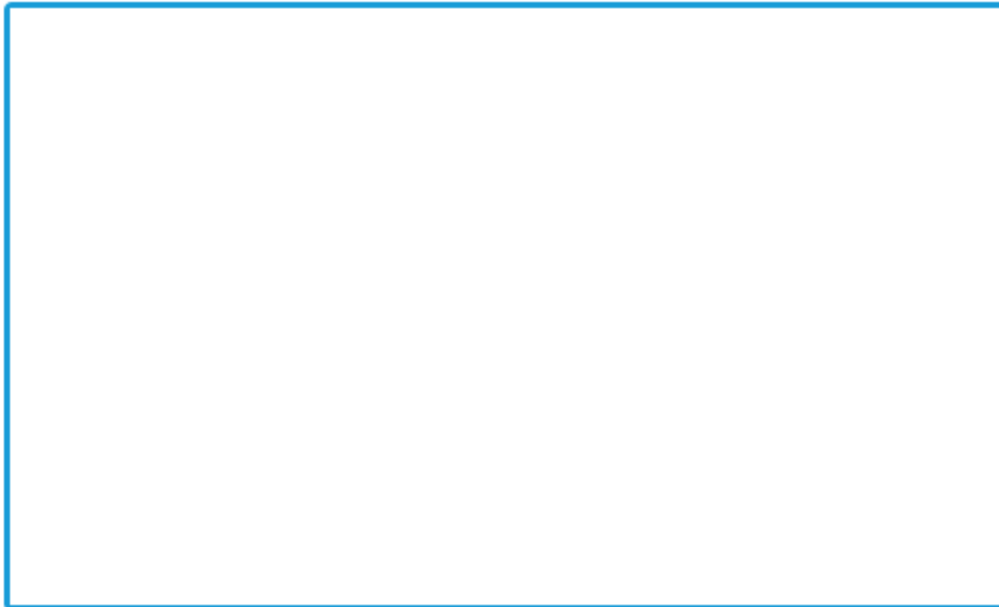
Create app.ts file and past the below content

**JS**

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module SignatureComponent {
  $(function () {
    var basicSignature = new ej.Signature($("#signature"), {
    });
  })
}
```

Now build your application, so that the **app.js** file is automatically generated and got added to your project (User have nothing to do with this file). Now, whatever code changes that you make in **app.ts** file will be reflected in app.js file automatically.

This will render a Signature like below.



### Adjusting Signature Size

You can customize the width and height of the Signature by **width** and **height** properties. These properties completely depend on signature canvas. The width and height are adjusted within the signature canvas.

The following code example is used to render the Signature component with customized width and height.

**JS**

```
<div id="signature" />
<script src="app.js"></script>
```

**JS**

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module SignatureComponent {
  $(function () {
    var basicSignature = new ej.Signature($("#signature"), {
      height: "300px",
      width: "200px",
      isResponsive: true
    });
  });
}
```

The following screenshot illustrates signature with customized width and height.



## Signature Customization

### Background color

Signature widget allows you to set the background Color for the control using the **backgroundColor** property. When we set our required background, then the signature's background color will be changed automatically.

The following code example is used to render the Signature control with background color

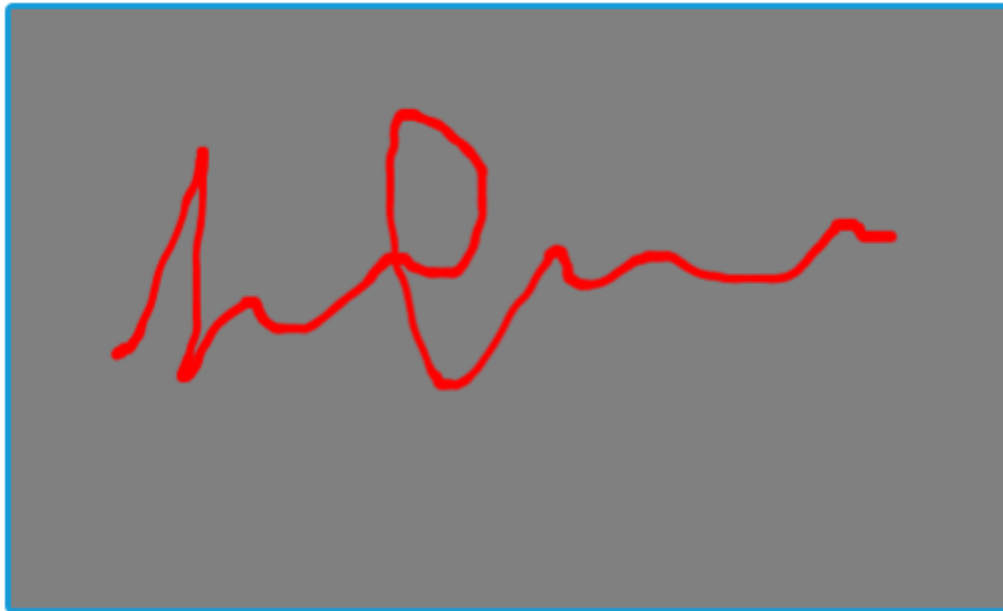
Add the following script to render signature with customized background color.

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SignatureComponent {
  $(function () {
    var basicSignature = new ej.Signature($("#signature"), {
      backgroundColor: "#808080"
    });
  });
}
```



The following screenshot illustrates the Signature with custom defined background color.



#### Background Image

Signature widget allows you to set the background image for the control using the **backgroundImage** property. When we set our required background image, then the signature's canvas will be changed with the given image.

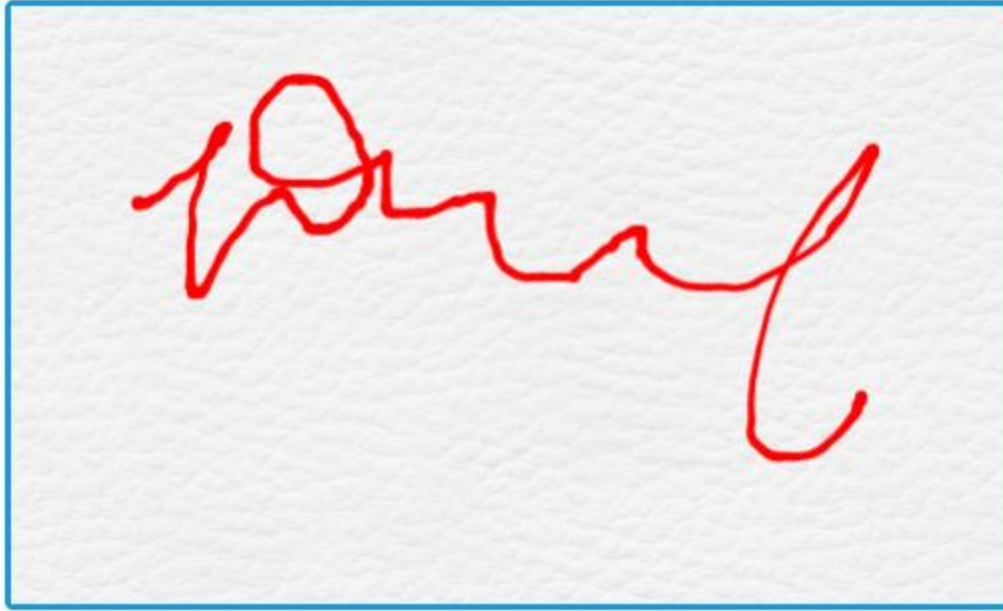
The following code example is used to render the Signature control with customized background image.

Add the following script to render signature with customized value.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SignatureComponent {
  $(function () {
    var basicSignature = new ej.Signature($("#signature"), {
      backgroundImage: "image.jpg"
    });
  });
}
```

The following screenshot illustrates the Signature with custom defined background image.



#### Stroke color

Signature widget allows you to set the stroke color for the signature stroke using the **strokeColor** property. When we set our required color, then the signature's stroke color will be changed.

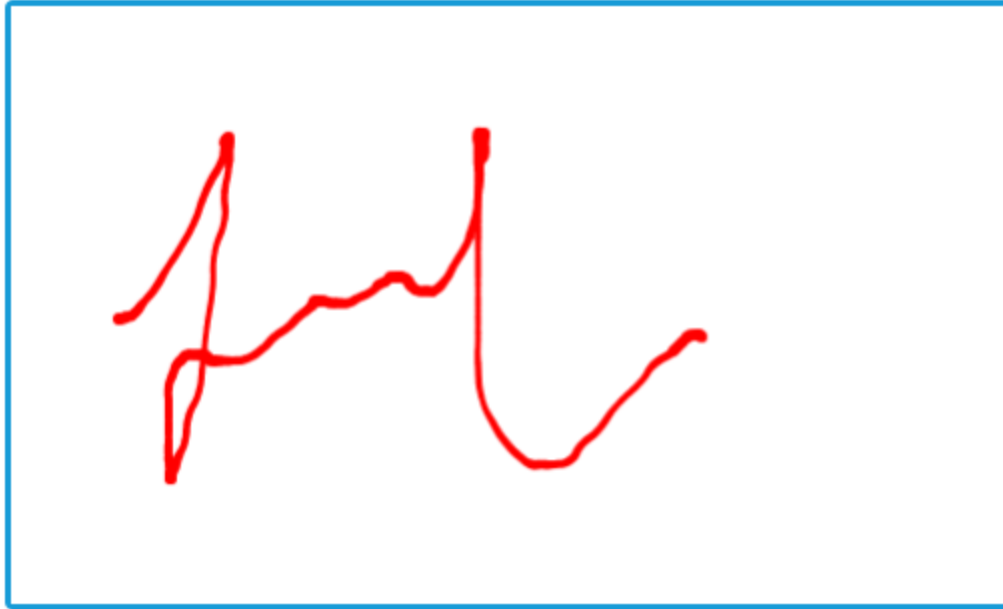
The following code example is used to render the Signature control with stroke color.

Add the following script to render signature with customized stroke color.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SignatureComponent {
  $(function () {
    var basicSignature = new ej.Signature($("#signature"), {
      strokeColor: "red"
    });
  });
}
```

The following screenshot illustrates the Signature with custom defined stroke color.



### Stroke Width

Signature widget allows you to set the stroke width for the control using the **strokeWidth** property. When we set our required width, then the signature's stroke width will be changed.

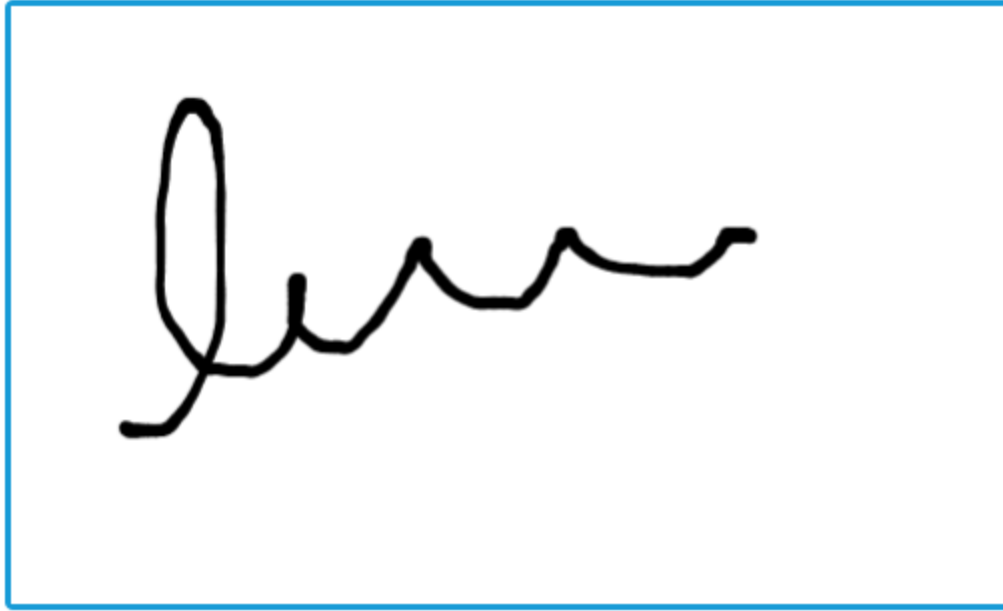
The following code example is used to render the Signature control with maximum stroke value.

Add the following script to render signature with customized stroke width.

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SignatureComponent {
  $(function () {
    var basicSignature = new ej.Signature($("#signature"), {
      strokeWidth: 4
    });
  });
}
```

The following screenshot illustrates the Signature with custom defined stroke maximum value.



## How To

### *Save signature image with user defined format*

By default, the downloaded image from the signature canvas will be in **png** format. We can define our own format to download the image with **saveImageFormat** property. And we can also save the image along with the background by using the **saveWithBackground** property.

The following code example is used to download drawn image on the Signature control.

### HTML

```
<div id="signature"></div>
<input id="save" type="button" value="save" />
```

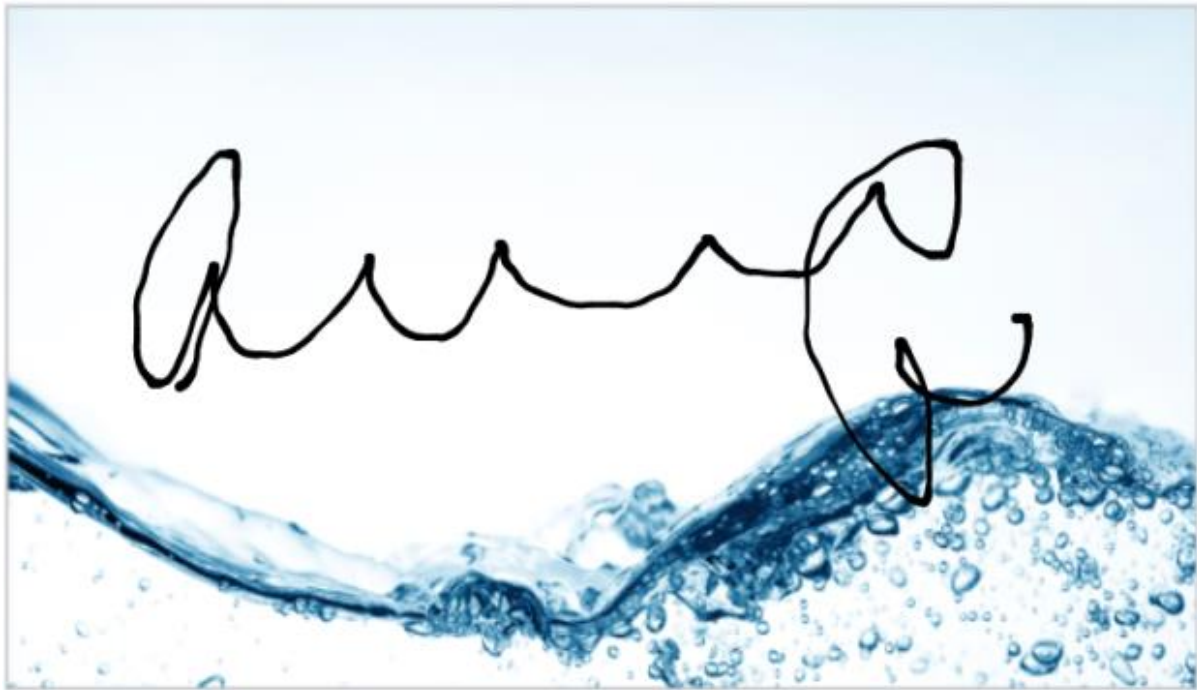
Add the following script to define the download format for the canvas

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SignatureComponent {
    $(function () {
        var basicSignature = new ej.Signature($("#signature"), {
            height: "500px",
            saveWithBackground: true,
            strokeWidth: 3,
            saveImageFormat: "jpg",
            backgroundImage: "../content/images/progressbar/water.png",
        });
        var button = new ej.Button($("#signSave"), {
            size: "normal", width: "70px",
            showRoundedCorner: true,
            click: onSave
        });
    });
}
```

```
function onSave(args) {  
  var signature = $("#signature").ejSignature("instance");  
  signature.save("MySignature");  
}
```

The following screenshot illustrates the Signature with saving (downloading) the drawn image.



#### *To clear the Signature*

To clear the signature, you can simply use the **clear()** method. This method will clear all the drawn strokes in the signature canvas and leaves it empty.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module SignatureComponent {  
  $(function () {  
    var basicSignature = new ej.Signature($("#signature"), {  
      height: "500px",  
      strokeWidth: 3  
    });  
    var button = new ej.Button($("#signatureClear"), {  
      size: "normal", width: "70px",  
      showRoundedCorner: true,  
      click: "onClear"  
    });  
  });  
  function onClear(args) {  
    var signature = $("#signature").ejSignature("instance");  
    signature.clear();  
  }  
}
```

### *Make signature as responsive*

When the signature control is resized or even the window is resized the strokes drawn in the signature will be disappeared. To make the strokes visible even after resizing the window, we must set the **isResponsive** property as true.

The following code example is used to render the Signature control with responsive support.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SignatureComponent {
  $(function () {
    var basicSignature = new ej.Signature($("#signature"), {
      isResponsive: true
    });
  });
}
```

The following screenshot illustrates the Signature with responsiveness.

Before Responsiveness:



After giving the Responsiveness:



*To check whether any input to the signature control since render*

We can detect whether or not there has been any input to the signature control since render. To detect we can use the `storeSnap` public variable, which is an array that stores all the canvas inputs. At initial rendering this array is empty and we can use this variable to check for the drawn strokes.

### JS

```
var signature = new ej.Signature($("#signature"), { });  
if (ej.isNullOrUndefined(signature.storeSnap)) {  
  //Something  
}
```

## Slider

### Overview

The **Essential JavaScript Slider** provides support to select a value from a particular range as well as selects a range value. The **Slider** has a sliding base on which the handles are moved. There are three types of **Sliders** such as default Slider, min-range Slider and range Slider.

### Key Features

- **Orientation** — Slider control is displayed in horizontal or vertical direction.
- **Animation** — Supports for handle movement with animation.
- **Tooltip** — Supports to display a tooltip to display the currently selected value.
- **Range Slider** — Supports to select a range of values.
- **Scale** — Supports to display a scale with small and big ticks.

- **Persist** — Supports for state maintenance while refreshing a page.
- **Customizable** — Easily customizable.
- **RTL Support** — Sets the alignment to the right, the reading order to right-to-left and the layout of the control to flow from right to left.

## Getting Started

Using the following steps, you can create a **TypeScript** Slider component.

### Creating an Slider in TypeScript

You can create a **TypeScript** application with the help of the given [TypeScript Getting Started Documentation](#).

Within an index.html file and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>TypeScript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/fl
at-azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.
web.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add Slider sample here-->
</body>
</html>
```

The Slider can be created from a HTML `div` element with the HTML `id` attribute and pre-defined options set to it.

#### HTML

```
<div id="minSlider"></div>
<script src="app.js"></script>
```

- Create app.ts file and use the below content

#### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#minSlider"), {
sliderType: "MinRange",
height: "16px",
width: "300px",
```



```
value: 60
});
});
}
```

- Now build your application, so that the **app.ts** file will be compiled and automatically generate the **app.js** file which is added to your project (User has nothing to do with this file). Now, whatever code changes that you make in **app.ts** file will be reflected in **app.js** file by compiling the application.

Execution of above code will render the following output.



set min and max values

- To set the maximum/ending value of the Slider, you can use the **maxValue** property. By default its value is 100. Data type of this property is "number".

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#minSlider"), {
sliderType: "MinRange",
height: "16px",
width: "300px",
value: 60,
maxValue: 50,
});
});
}
```

- To set the minimum/starting value of the Slider, you can use the **minValue** property. By default its value is 0. Data type of this property is "number".

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#minSlider"), {
sliderType: "MinRange",
height: "16px",
width: "300px",
value: 60,
minValue: 50,
});
});
}
```

```
});  
});  
}
```

## Behavior Settings

### Height

By default, **Slider** renders with a height of 14px. You can change the **Slider** height using **height** property. You can specify the value for this property in number or string format.

### Width

By default, **Slider** widget renders with 100% width. You can customize the width of the Slider using **width** property. You can specify the value for this property in number or string format.

The following steps explain you on how to configure the **height** and **width** of the **Slider**.

In a HTML page, add a <div> element to render it as a Slider widget.

### HTML

```
<div id="BasicSlider"></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module SliderComponent {  
    $(function () {  
        var slider = new ej.Slider($("#BasicSlider"), {  
            height: "20",  
            width: "500"  
        });  
    });  
}
```

Execute the above code example to render the following output.



### IncrementStep

This property sets the incremental step value for the **Slider**. When the **Slider** handle slides through mouse or keyboard, it increments / decrements the value based on the step value. By default, when the slider handle is moved, the increments / decrements value is one. Using **incrementStep** property you can change the increment step value. Data type of this property is number.

The following steps explain you on how to configure the **incrementStep** property.

In a **HTML** page, add a <div> element to render it as a **Slider** widget.

### HTML

```
<div id="ejSlider"></div>
```

### JAVASCRIPT

```
// When initializing the Slider widget, configure the "incrementStep"
property as follows.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
  $(function () {
    var slider = new ej.Slider($("#ejSlider"), {
      height: "15",
      width: "500",
      incrementStep: 5
    });
  });
}
```

Execute the above code example to render the following output.



In the above example, value for **incrementStep** property is specified as "5" therefore, when you move the **Slider** handle, value "5" increments/decrements from the current **Slider** value.

### ReadOnly

This feature prevents you from interacting with the **Slider**. That is you can only view the **Slider** value and cannot change it.

The following steps explain you on how to enable the **readOnly** property.

In a **HTML** page, add a **<div>** element to render it as a **Slider** widget.

### HTML

```
<div id="ejSlider"></div>
```

### JAVASCRIPT

```
// When initializing the Slider widget, enable the "readOnly" property as
follows.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
  $(function () {
    var slider = new ej.Slider($("#rangeSlider"), {
      height: "15",
      width: "500",

```

```
readOnly:true
});
});
}
```

After you execute the above code example, the **Slider** values cannot be changed in any ways.

### Persistence Support

This feature supports you to save current model value to browser cookies for state maintenance. When you refresh the web page, the **Slider** control retains the model value apply from browser cookies. The data type of **enablePersistence** is Boolean type.

The following steps explain you on how to enable the **enablePersistence** property.

In an **HTML** page, add a **<div>** element to render it as a **Slider** widget.

#### HTML

```
<div id="ejSlider"> </div>
```

#### JAVASCRIPT

```
// When initializing the Slider widget, enable the enablePersistence
// property as follows.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#ejSlider"), {
height: "15",
width: "500",
enablePersistence:true
});
});
}
```

### Orientation

This property is used to set the **Slider** in either horizontal or vertical direction. By default, **Slider** renders in horizontal direction. Data type of this property is “enum”.

Possible **Slider** orientations are as follows,

Property Table for JavaScript

| Property    | Allowed values                            | Description                                 |
|-------------|---|---|
| orientation | ej.Orientation.Vertical                   | Displays the Slider in vertical direction   |
|             | ej.Orientation.Horizontal (default value) | Displays the Slider in horizontal direction |

The following steps explains you on how to configure the **orientation** property.

In an **HTML** page, add a **<div>** element to render it as a **Slider** widget.

## HTML

```
<div id="ejSlider"></div>
```

## JAVASCRIPT

```
// When initializing the Slider widget, configure the orientation
property as follows.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#ejSlider"), {
height: "150",
width: "20",
orientation:ej.Orientation.Vertical
});
});
}
```

Execute the above code example to render the following output.



## Slider Types

This feature allows you to specify the type of **Slider**. There are three different types of **Slider**, **Default Slider**, **Min-Range Slider** and **Range Slider**. By default, **Default Slider** renders. You can use the **sliderType** property to choose the type of **Slider**. Data type of this property is “Enum”

Both **Default Slider** and **Min-Range Slider** have same behavior that is used to select a single value. In Min-Range Slider, a shadow is considered from the start value to current handle position. But Range Slider contains two handles that is used to select a range of values and a shadow is considered in between the two handles.

Possible Slider types are as follows,

Property Table for JavaScript

| Property   | Allowed values                        | Description   |
|------------|---------------------------------------|---|
| sliderType | ej.SliderType.Default (default value) | It is the default Slider type. It helps to select a single value. |

|  |                        |  |
|--|------------------------|--|
|  | ej.SliderType.MinRange | Use this Slider to select a single value; Displays shadow from the start value to the current value. |
|  | ej.SliderType.Range    | Use this Slider to select a range of values; Displays shadow in-between the selection range.         |

The following steps explain you on how to configure the **sliderType** property to display **Range Slider** and **MinRange Slider**.

In an **HTML** page, specify the **<div>** elements to render the **RangeSlider** and **MinRange Slider**.

#### HTML

```
<div class="txt">Range</div>
<div id="rangeSlider"></div>
<br />
<br />
<div class="txt">Min-Range</div>
<div id="minSlider"></div>
```

#### JAVASCRIPT

```
// When initializing the Slider components, configure the sliderType
property as follows.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#minSlider"), {
sliderType: ej.SliderType.MinRange,
value: 60,
width:"500"
});
var slider1 = new ej.Slider($("#rangeSlider"), {
sliderType: ej.SliderType.Range,
values: [30, 60],
width:"500"
});
});
});
}
```

Execute the above code example to render the following output.

Range



Min-Range



## Updating slider value

**Slider** widget includes an option to specify/update its value. And also you can specify the starting and ending value for the **Slider** using the **minValue** and **maxValue** properties.

### Value

This property is used to set the value in the “**Default**” and “**Min-Range**” Sliders. By default its value is null when no value is specified. Data type of this property is “**number**”.

You can get/set the value in the slider handle by using [getValue](#) and [setValue](#) methods.

Also [change](#) event will be triggered whenever **Slider** value is changed.

### Values

This property is used to set the value in “**Range Slider**”. By default range values is from 0 to 100. This property is of “**Array**” data type.

The following steps explains you the configuration of “**value**” and “**values**” property.

In an **HTML** page, specify the **<div>** elements to render the “**Range Slider**” and “**Default Slider**”.

### HTML

```
<div class="txt">Default Slider</div>
<div id="defaultSlider"></div>
<br />
<br />
<div class="txt">Range Slider</div>
<div id="rangeSlider"></div>
```

### JAVASCRIPT

```
// When initializing Slider widget, configure the "value" and "values"
// property as follows.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
    $(function () {
        var slider = new ej.Slider($("#minSlider"), {
            sliderType: ej.SliderType.Default,
            value: 50,
            width: "500"
        });
        $("#rangeSlider").ejSlider({
            sliderType: ej.SliderType.Range,
            values: [20, 80],
            width: "500"
        });
    });
}
```

Execute the above code example to render the following output.

Default Slider



Range Slider



### MinValue

To set the minimum/starting value of the Slider, you can use the `minValue` property. By default its value is 0. Data type of this property is **"number"**.

### MaxValue

To set the maximum/ending value of the Slider, you can use the `maxValue` property. By default its value is 100. Data type of this property is **"number"**.

The following steps explain you on how to configure `minValue` and `maxValue` property.

In an **HTML** page, specify the **<div>** elements to render the **Default Slider** and **Range Slider**.

### HTML

```
<div class="txt">Default Slider</div>
<div id="defaultSlider"></div>
<br />
<br />
<div class="txt">Range Slider</div>
<div id="rangeSlider"></div>
```

### JAVASCRIPT

```
// When initializing Slider widget, configure the minValue and maxValue
properties as follows.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#defaultSlider"), {
sliderType: ej.SliderType.Default,
value: 60,
width: "500",
minValue: 40,
maxValue: 80
});
var slider = new ej.Slider($("#rangeSlider"), {
sliderType: ej.SliderType.Range,
values: [10, 90],
width: "500",
minValue: 10,
maxValue: 90
});
});
});
}
```

Execute the above code example to render the following output.



Default Slider



Range Slider



In the above example, for **Default Slider** the slider value starts from “40” (min value) and ends in “80” (max value), so the slider handle is placed at the center of the Slider while specifying the value as “60”.

For **Range Slider**, the value starts from “10” (min value) and ends in “90” (max value). The range shadow occupies the entire **Slider**, since the range (values) is specified as “[10, 90]”.

### Buttons

**Slider** includes the button support for increment or decrement the values of the slider.

#### Enabling Buttons

Use the **showButtons** property to enable the button support. By default this property is disabled. Data type of this property is “Boolean”.

The following steps explain you on how to enable button support in **Slider**.

In an **HTML** page, specify the **<div>** elements to render the **Range Slider**.

### HTML

```
<div class="txt">Range Slider</div>
<div id="rangeSlider"></div>
```

### JAVASCRIPT

```
// In JavaScript, when initializing the Slider, specify the value for
"showButtons" property as "true"
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#rangeSlider"), {
sliderType: ej.SliderType.Range,
values: [30, 60],
width: "500",
showButtons: true
});
});
}
```

Execute the above code example to render the following output.

Range



## Scale Settings

**Slider** widget includes option to display the scale on the **Slider**. Scale in **Slider** supports you to easily identify the current value/values of the **Slider**. Scale contains “small ticks” and “large ticks”. Values of the **Slider** is displayed above each large ticks.

### Show Scale

This property enables the scale in the **Slider**. By default, its value is “false”. Data type of this property is “Boolean”.

The following steps explains you the configuration of **showScale** property.

In an **HTML** page, specify the **<div>** elements to render the “Default Slider” and “Range Slider”.

### HTML

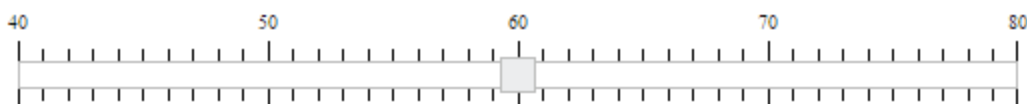
```
<div class="txt">Default Slider</div>
<div id="defaultSlider"></div>
<br />
<br />
<div class="txt">Range Slider</div>
<div id="rangeSlider"></div>
```

### JAVASCRIPT

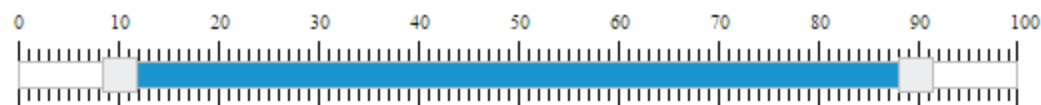
```
// In JavaScript, configure the showScale property as follows.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
  $(function () {
    var slider = new ej.Slider($("#defaultSlider"), {
      sliderType: ej.SliderType.Default,
      value: 60,
      width: "500",
      minValue: 40,
      maxValue: 80,
      showScale: true
    });
    var slider1 = new ej.Slider($("#rangeSlider"), {
      sliderType: ej.SliderType.Range,
      values: [10, 90],
      width: "500",
      showScale: true
    });
  });
}
```

Execute the above code example to render the following output.

### Default Slider



### Range Slider



#### Enable Small Ticks

**Slider** widget provides you an option to enable/disable the small ticks present in the scale. By default, when you enable “showScale” property, small ticks is displayed in the scale. Use the **showSmallTicks** property to disable the small ticks present in the scale. Data type of this property is “Boolean”. The [renderingTicks](#) event will be triggered while creating each slider scale tick.

The following steps explains you on how to disable the small ticks in **Slider**.

In an **HTML** page, specify the **<div>** elements to render the “Default Slider” and “Range Slider”.

#### HTML

```
<div class="txt">Default Slider</div>
<div id="defaultSlider"></div>
<br />
<br />
<div class="txt">Range Slider</div>
<div id="rangeSlider"></div>
```

#### JAVASCRIPT

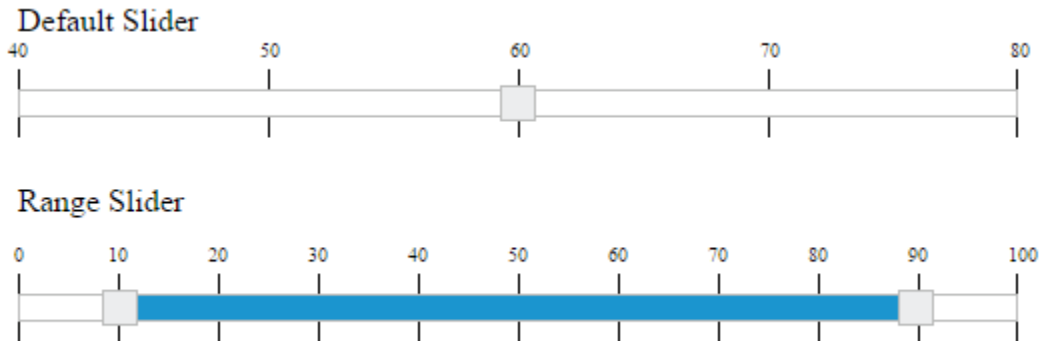
```
// In JavaScript, set the value of showSmallTicks property as "false".
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#defaultSlider"), {
sliderType: ej.SliderType.Default,
value: 60,
width: "500",
minValue: 40,
maxValue: 80,
showScale: true,
showSmallTicks: false
});
var slider = new ej.Slider($("#rangeSlider"), {
sliderType: ej.SliderType.Range,
values: [10, 90],
width: "500",
showScale: true,
showSmallTicks: false
});
});
}
```

```

    });
  });
}

```

Execute the above code example to render the following output.



#### Small step

This property specifies the distance between the two small ticks present in the scale and the position of the small ticks in the Slider scale. Data type of this property is “number”.

#### Large step

This property specifies the distance between the two small ticks present in the scale and the position of the large ticks in the Slider scale. Data type of this property is “number”.

The following steps explain you on how to configure the smallStep and largeStep property in Slider scale.

In an **HTML** page, specify the **<div>** elements to render the “Default Slider” and “Range Slider”

#### HTML

```

<div class="txt">Default Slider</div>
<div id="defaultSlider"></div>
<br />
<br />
<div class="txt">Range Slider</div>
<div id="rangeSlider"></div>

```

#### JAVASCRIPT

```

// In JavaScript, specify the value for smallStep and largeStep
properties.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
  $(function () {
    var slider = new ej.Slider($("#defaultSlider"), {
      sliderType: ej.SliderType.Default,
      value: 60,
      width: "500",
      minValue: 40,
      maxValue: 80,

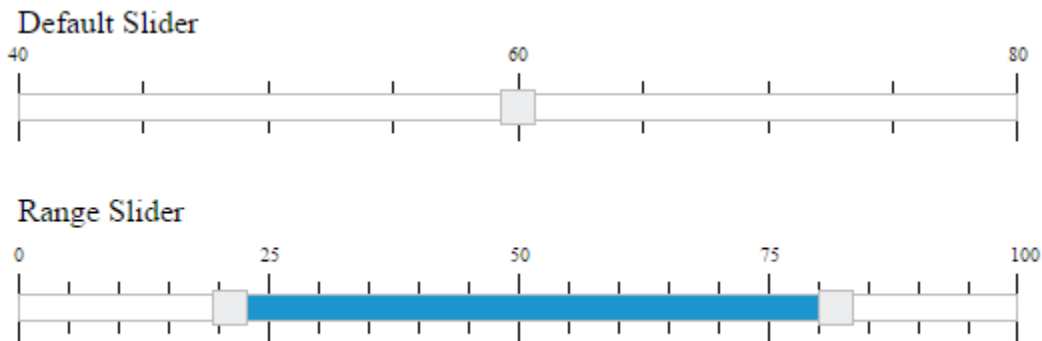
```

```

showScale: true,
smallStep: 5,
largeStep: 20
});
var slider1 = new ej.Slider($("#rangeSlider"), {
sliderType: ej.SliderType.Range,
values: [25, 75],
width: "500",
showScale: true,
smallStep: 5,
largeStep: 25
});
});
}

```

Execute the above code example to render the following output.



In the above example, for “Default Slider” the “**smallStep**” value is specified as “5”, so for each 5 values from the starting value, small ticks is enabled. Also, “**largeStep**” value is specified as “20”, so for each 20 values from the starting value, large ticks is enabled.

Similarly for “Range Slider”, “**smallStep**” value is specifies as “5”, so for each 5 values from the starting value, small ticks is enabled. Also, “**largeStep**” value is specified as “25” so, for each 25 values large ticks is enabled.

### Appearance and Styling

**Slider** widget looks sleek and enriched with good UI appearance. It is included with both metro (flat) theme and gradient theme support. Totally twelve built-in themes are provided including six flat themes and six gradient themes. The themes include three color variations such as “Azure”, “Lime” and “Saffron”. The themes supported by the **Slider** widgets are as follows,

By default, there are 16 theme support available for RadialMenu component, namely:

- default-theme
- flat-azure-dark
- flat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure

- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap
- high-contrast-01
- high-contrast-02
- material
- office-365

In order to apply different themes, you can refer the “**ej.widgets.all.min.css**” file from the corresponding theme folders. This file is a combination of two style sheets “**ej.widgets.core.min.css**” and “**ej.theme.min.css**”. Instead of including “**ej.widgets.all.min.css**” file you can also refer the “**ej.widgets.core.min.css**” and “**ej.theme.min.css**” files separately.

The following steps explains you on how to apply “**flat-lime-dark**” theme to the **Slider** widget

In an **HTML** page, specify the desired “**ej.widgets.all.min.css**” file to load the corresponding theme.

### HTML

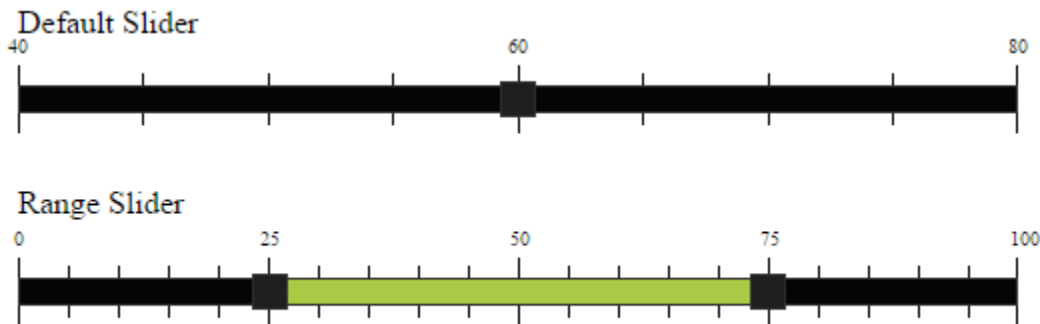
```
<!--In _Layout page, specify the desired "ej.widgets.all.min.css" file to load the corresponding theme.-->
<head>
<title>Slider</title>
<!--Flat-Lime theme-->
<link href="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/flat-lime-dark/ej.web.all.min.css"rel="stylesheet"/>
<!--scripts-->
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-1.10.2.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/ej.web.all.min.js"></script>
</head>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#defaultSlider"), {
sliderType: ej.SliderType.Default,
value: 60,
width: "500",
minValue: 40,
maxValue: 80,
showScale: true,
smallStep: 5,
largeStep: 20
});
var slider = new ej.Slider($("#rangeSlider"), {
sliderType: ej.SliderType.Range,
values: [25, 75],
```

```
width: "500",
showScale: true,
smallStep: 5,
largeStep: 25
});
});
}
```

Execute the above code example to render the following output.



### CSS Class

When you want to display the **Slider** widget in a different style based on the appearance of your application, you can use this **cssClass** property to apply custom theme for the **Slider**. Specify a class name as the value for **cssClass** property. The specified class is added to the wrapper of the **Slider** widget. Now, you can easily override the styles of the **Slider** widget by accessing the styles from the root level (using the **cssClass** specified).

The following steps explain you on how to configure the **Slider** with custom theme using the **cssClass** property. Here, a class name "purple" is specified for the **cssClass**.

In an **HTML** page, specify the **<div>** elements to render the "Default Slider" and "Range Slider".

### HTML

```
<div class="txt">Range Slider</div>
<div id="rangeSlider"></div>
```

### JAVASCRIPT

```
// In JavaScript, specify a class as the value for "cssClass" property.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#rangeSlider"), {
sliderType: ej.SliderType.Range,
values: [25, 75],
width: "500",
cssClass: "purple"
});
});
}
```

Include the **cssClass** value before each style of the **Slider** widget and customize the styles as follows.

### CSS

```
<style>
.purple.e-slider.e-widget {
background-color: gray;
border-color: #bbbcbb;
}
.purple.e-tooltip {
background: none repeat scroll 0 0 violet;
/* Old browsers */
border-color: #1b95cf;
color: white;
}
.purple.e-slider .e-handle.e-select {
background-color: purple;
border-color: purple;
}
.purple.e-slider .e-handle.e-hover {
background-color: purple;
border-color: purple;
}
.purple.e-slider .e-handle.e-focus {
box-shadow: 0 0 2px rgba(0, 0, 0, 0.2);
}
.purple.e-slider .e-range {
background: none repeat scroll 0 0 violet;
/* Old browsers */
}
.purple.e-scale .e-tick {
background-image: url(images/dot.png);
}
</style>
```

Execute the above code example to render the following output.

### Range Slider



### Show Tooltip

**Slider** displays the tooltip to indicate the current value when you click on the **Slider** handle. By default, **Slider** displays the tooltip. Using the **showTooltip** option you can enable or disable the Tooltip. Data type of this property is “boolean”.

The following steps explain you on how to disable the tooltip in **Slider**.

In an **HTML** page, specify the **<div>** elements to render the **Default Slider**.



## HTML

```
<div class="txt">Default Slider</div>
<div id="defaultSlider"></div>
```

## JAVASCRIPT

```
// In JavaScript, during initialization disable the showTooltip property.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#defaultSlider"), {
value: 60,
width: "500",
showTooltip:false
});
});
}
```

### Show Rounded Corner

This property is used to display the **Slider** and its handle with rounded corners. By default **showRoundedCorner** is in disabled state. Data type of this property is "Boolean".

The following steps explains you on how to disable the tooltip in **Slider**.

In an **HTML** page, specify the **<div>** elements to render the "Default Slider".

## HTML

```
<div class="txt">Default Slider</div>
<div id="defaultSlider"></div>
```

## JAVASCRIPT

```
// In JavaScript, during initialization enable the showRoundedCorner
property.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#defaultSlider"), {
value: 60,
width: "500",
showRoundedCorner:true
});
});
}
```

Execute the above code example to render the following output.

## Default Slider



## Animation

This feature includes the animation effect for the **Slider** when moving the **Slider** handle.

### Enabling Animation

By default, animation is enabled in the **Slider**. Using the **enableAnimation** property you can enable/disable the animation effects. Data type of this property is “Boolean”.

The following steps explain you on how to disable the animation effect in **Slider**.

In an **HTML** page, specify the **<div>** elements to render the “Default Slider”.

### HTML

```
<div class="txt">Default Slider</div>
<div id="defaultSlider"></div>
```

### JAVASCRIPT

```
// In JavaScript, during initialization disable the enableAnimation
property.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
  $(function () {
    var slider = new ej.Slider($("#defaultSlider"), {
      value: 60,
      width: "500",
      enableAnimation: false
    });
  });
}
```

## Customizing Animation speed

Animation speed of the **Slider** indicates the speed at which the slider handle can be moved. Higher the value specified for this property decreases the rate of speed at which the **Slider** handle can be moved. You can customize the animation speed of the **Slider** using the **animationSpeed** property. Default value of this property is “500”.

The following steps explain you on how to customize the animation speed.

In an **HTML** page, specify the **<div>** elements to render the “Default Slider”.

### HTML

```
<div class="txt">Default Slider</div>
<div id="defaultSlider"></div>
```

### JAVASCRIPT

```
// In JavaScript, during initialization specify the value for
animationSpeed property.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#defaultSlider"), {
value: 60,
width: "500",
animationSpeed: 600
});
});
}
```

## Enable/Disable the Slider

**Slider** widget includes an option to enable/disable it. When you disable the Slider, it is displayed in a blur state and you cannot perform any operations in it.

### Enabled

Using this **enabled** property you can enable/disable the **Slider**. Data type of this property is "Boolean". Also you can enable/disable the **Slider** by using [enable](#) and [disable](#) methods.

The following steps explains you on how to disable the **Slider**.

In an **HTML** page, specify the **<div>** elements to render the **Default Slider**.

### HTML

```
<div class="txt">Default Slider</div>
<div id="defaultSlider"></div>
```

### JAVASCRIPT

```
// During initialization disable the Slider by setting value for enabled
property as false.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#minSlider"), {
value: 60,
width: "500",
enabled: false
});
});
}
```

Execute the above code example to render the following output.

## Default Slider



## RTL support

**Slider** includes the Right to Left alignment support. Operations in the **Slider** is performed from Right to Left.

### Enabling RTL

Use the **enableRTL** property to enable the RTL support. By default this property is disabled. Data type of this property is "Boolean".

The following steps explains you on how to enable RTL support in **Slider**.

In an **HTML** page, specify the **<div>** elements to render the **Range Slider**.

### HTML

```
<div class="txt">Range Slider</div>
<div id="rangeSlider"></div>
```

### JAVASCRIPT

```
// In JavaScript, when initializing the Slider, specify the value for
"enableRTL" property as "true"
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#rangeSlider"), {
sliderType: ej.SliderType.Range,
values: [25, 75],
width: "500",
enableRTL: true
});
});
}
```

Execute the above code example to render the following output.

## Range Slider



## Keyboard Interaction

You can use Keyboard shortcut keys as an alternative to the mouse for using the **Slider** widget. All options in the **Slider** can be accessed using keyboard shortcuts. The following table explains the keyboard shortcut keys and the operations that can be performed using the corresponding keys.

List of Keyboard shortcuts

| Shortcut Key | Description                            |
|--------------|--|
| Alt + j      | Focuses into the Slider handle         |
| Up/Right     | Increments the Slider value            |
| Down/Left    | Decrements the Slider value            |
| Home         | Slider handle moves to the start value |
| End          | Slider handle moves to the end value   |
| Esc          | Focuses out from the Slider handle     |

## Configure keyboard interaction

The following steps explain you on how to enable keyboard support in **Slider**.

In an **HTML** page, specify the **<div>** elements to render the **Range Slider**.

### HTML

```
<div class="txt">Range Slider</div>
<div id="rangeSlider"></div>
```

### JAVASCRIPT

```
// Focus the Slider control in the document key down function.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SliderComponent {
$(function () {
var slider = new ej.Slider($("#rangeSlider"), {
sliderType: ej.SliderType.Range,
values: [25, 75],
width: "500",
enableRTL: true
});
$(document).on("keydown", function (e) {
if (e.altKey && e.keyCode === 74) { // j- key code.
$("#rangeSlider a")[0].focus();
}
});
});
}
```

Run the sample and press, **Alt + j** keys to set focus in the **Slider** handle and you can handle the Slider operations using the keyboard shortcut keys.

## Sparkline

### Overview

Sparklines are easy to interpret and also it conveys much more information to the user by visualizing the data in a small amount of space.

Some of the key features are,

- Consumes less amount of space for visualizing the data.
- Easy integration with grids.
- Five type of Sparkline available namely Line, Column, Area, Win-loss and Pie.
- Range Band to highlight selected start and end value region.
- Can customize markers for high, low, start, end and negative point.
- Trackball to get current mouse point to display tooltip.

### Getting Started

This section explains you the steps required to populate the Sparkline with data, tooltips and type for Sparkline. This section covers only the minimal features that you need to know to get started with the Sparkline.

#### Create your sparkline

You can easily create the Sparkline widget by using the following steps.

1.First create an TypeScript Project and add the following script reference in the app.ts page

For common getting started of TypeScript , you can refer [here](#).

The default type definition file **ej.web.all.d.ts** needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the **jquery.d.ts** file in your TypeScript application, which can be downloaded from [here](#).

2.Add the following script reference in the HTML page

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
```

```
</body>
</html>
```

In the above code, `ej.web.all.min.js` script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

3. Create a

tag.

#### HTML

```
<html> <body> <div id="Sparkline"></div> </body> </html>
```

4. Initialize the Sparkline in ts file by using the `ej.Sparkline` method.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SparklineComponent {
  $(function () {
    var sample = new ej.datavisualization.Sparkline($("#Sparkline"));
  });
}
```

Now, the Sparkline is rendered with some auto-generated random values and with default Line type Sparkline.



#### Simple Sparkline

The Sparkline is rendered to its default size. You can also customize the Sparkline dimension either by setting the width and height of the container element as in the above code example or by using the `size` of the Sparkline.

#### Populate Sparkline with data

Now, this section explains how to plot JSON data to the Sparkline. First, let us prepare a sample JSON data with each object containing following fields – `employeeId` and `sales`.

```
var sparklineData = [
  { employeeId: 1, sales: 25 },
  { employeeId: 2, sales: 28 },
  { employeeId: 3, sales: 34 },
  { employeeId: 4, sales: 32 },
  { employeeId: 5, sales: 40 },
```

```
{ employeeId: 6, sales: 32 },  
{ employeeId: 7, sales: 35 },  
{ employeeId: 8, sales: 55 },  
{ employeeId: 9, sales: 38 },  
{ employeeId: 10, sales: 30 }];
```

Now, add the dataSource to the Sparkline and provide the field name to get the values from the dataSource in xName and yName options.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
    dataSource: sparklineData,  
    xName: "employeeId",  
    yName: "sales",  
  });  
});
```

#### Sparkline Type

To change the sparkline types, following example code illustrates this.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
    type: "column"  
  });  
});
```

#### Enable Tooltip

To enable the Sparkline tooltip we can see the current point values.

The following code example illustrates enable tooltip in sparkline,

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
    tooltip: {  
      visible: true,  
    },  
  });  
});
```





## Working with Data

## Local Data

1. You can bind the data to the Sparkline by using `dataSource` property and then you need to map the X and Y value with `xName` and `yName` properties respectively.

**JAVASCRIPT**

```
var sparkLineData = [
  { employeeId: 1, sales: 25 },
  { employeeId: 2, sales: 28 },
  { employeeId: 3, sales: 34 },
  { employeeId: 4, sales: 32 },
  { employeeId: 5, sales: 40 },
  { employeeId: 6, sales: 32 },
  { employeeId: 7, sales: 35 },
  { employeeId: 8, sales: 55 },
  { employeeId: 9, sales: 38 },
  { employeeId: 10, sales: 30 }];
module linesparkline {
  $(function () {
    //...//
    var sample = new ej.Sparkline($("#line"), {
      dataSource: sparkLineData,
      xName: "employeeId",
      yName: "sales",
      //...//
    });
  });
}
```



2. You can also bind an array of data to Sparkline by using `dataSource` property.

**JAVASCRIPT**

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module linesparkline {
  $(function () {
    //...//
    var sample = new ej.Sparkline($("#line"), {
      dataSource: [12, 14, 11, 12, 11, 15, 12, 10, 11, 12, 15, 13, 12, 11, 10,
        13, 15, 12, 14, 16, 14, 12, 11],
      //...//
    });
  });
}
```



### Sparkline Dimensions

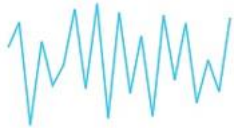
You can set the size directly on the Sparkline or to the container of the sparkline. When you do not specify the size, it takes 30px as the height and 50px as its width, by default.

#### Set size for the container

You can customize the Sparkline dimension by setting the width and height for the container element.

### HTML

```
<body>
<div id="container" style="width:820px;height:500px;"></div>
<script type="text/typescript" language="typescript ">
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"));
});
</script>
</body>
```



#### Set size in pixels

You can also set the **width** and **height** Sparkline by using the **size** property of the Sparkline.

### JAVASCRIPT

```
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
// ...
size: { width: '60', height: '40' },
// ...
});
});
```

### Responsive Sparkline

To resize the Sparkline when the browser or the sparkline container is resized, set the **isResponsive** property to **true**, where the sparkline adapts to the changes in size of the container.

### JAVASCRIPT

```
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
// ...
//Enable isResponsive to change the sparkline size dynamically.
isResponsive: true
});
});
```

```
// ...
} ) ;
} ) ;
```

## Sparkline Types

### Line Type

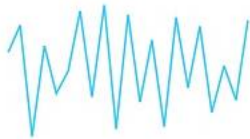
To render a Line type Sparkline, set the **type** as **line**. To change the color and width of the line, you can use the **fill** and **width** property.

# JAVASCRIPT

```

    
    /// <reference path="tsfiles/jquery.d.ts" />
    /// <reference path="tsfiles/ej.web.all.d.ts" />
    module SparklineComponent {
    $(function () {
    var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
    // ...
    width: 3,
    fill: "#33ccff",
    // ...
    });
    });
    }
    

```



## Column Type

To render a Column Sparkline, set the type as **column** To change the color of the column, you can use the **fill** property.

## JAVASCRIPT

```
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
// ...
type: 'column',
fill: "#33ccff",
// ...
});
});
```



### Area Type

To render an Area Sparkline, you can specify the type as **area**. To change the Area color, you can use the **fill** property

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
    // ...  
    type: 'area',  
    fill: '#69D2E7'  
    // ...  
  });  
});
```

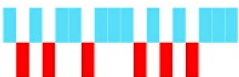


### WinLoss Type

WinLoss Sparkline render as a column segment and it show the positive, negative and neutral values. You can customize the positive and negative color of the win-loss type.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
    // ...  
    type: 'winLoss',  
    fill: '#69D2E7'  
    // ...  
  });  
});
```



### Pie Type

You can create a pie type sparkline by setting the type as **pie**. Colors for the pie can be customize using **palette** property.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
    // ...  
    type: 'pie',  
    palette: ['#ff3399', '#33ccff'],  
    // ...  
  });  
});
```



### Axis Customize

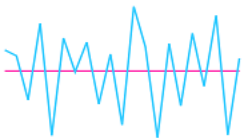
The Sparkline axis can be collapsed using visible property in **axisLineSetting** and this not applicable for win-loss. You can customize **color**, **width** and **dash array** of axis line.

#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SparklineComponent {
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
// ...
axisLineSettings: {
visible: true,
color: "#ff14ae",
}
// ...
});
});
}

```



### Marker Customization

You can customize markers by initializing the **markerSettings** property. The customization options allow you to change the **visibility fill, width, opacity** and **border** options **color, width, opacity** for marker. This customization only applicable for line, column and area type Sparkline.

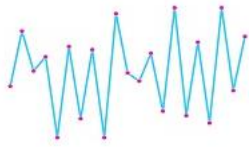
#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SparklineComponent {
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
// ...
markerSettings: {
visible: true,
width: 4,
fill: "#ff14ae",
border: {
width: 1
}
}
// ...
});
}

```

```
});
}
```



### Point Customization

You can customize points by initializing the point colors. The customization options allow you to differentiate the **first**, **last**, **highest**, **lowest**, and **negative** points. This customization only applicable for line, column and area type Sparkline.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SparklineComponent {
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
// ...
// ...
negativePointColor: "red",
highPointColor: "blue",
lowPointColor: "orange",
startPointColor: "green",
endPointColor: "green",
// ...
});
});
}
```



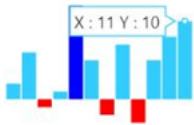
### Tooltip

A **tooltip** follows the pointer movement and is used to indicate the value of a point. This feature is applicable for line, column, pie, and area Sparkline. You can enable the tooltip by setting its **visible** property as **true**.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SparklineComponent {
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
// ...
tooltip: {
visible: true,
},
},
}
```

```
// ...
});
});
}
```

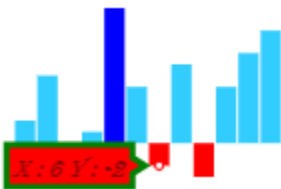


### Tooltip Customization

You can customize the tooltip fill color, border, border properties border color, border width and font properties color, font family, font style, font weight, opacity, size

#### JAVASCRIPT

```
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
// ...
tooltip: {
fill: 'red',
border: {
color: "green",
width: 3
},
font: {
size: "12px",
fontFamily : "Algerian",
fontStyle : "italic",
fontWeight : "lighter",
opacity : 0.5,
}
},
// ...
});
});
```



### Tooltip Template

HTML elements can be displayed in the tooltip by using the `template` option of the tooltip. The `template` option takes the value of the `id` attribute of the HTML element. You can use the `#point.x#` and `#point.y#` as place holders in the HTML element to display the x and y values of the corresponding point.

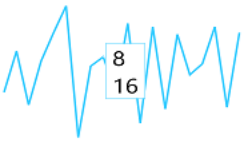
#### JAVASCRIPT

```
<div id="item" style="display: none;">
<div>
<div>#point.x#</div>
```

```

<div>#point.y#</div>
</div>
</div>
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
// ...
tooltip: {
visible: true,
template: "item",
},
// ...
});
});

```



### Range Band

The **range band** feature is used to highlight a particular range along the y-axis using **start** and **end** range property. You can also customize the **color** and **opacity** of the range band.

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SparklineComponent {
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
// ...
rangeBandSettings: {
startRange: 4,
endRange: 30,
color: "#ff14ae",
opacity: 0.4
},
// ...
});
});
}

```



### Sparkline Customization

This section explains you the customization options available to make changes in the Sparkline for getting better appearance.



### Sparkline background

You can specify the background color for the Sparkline using **background** property. When you don't specify the **background**, it takes "transparent" as background color.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SparklineComponent {
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
// ...
//Specifies background color for sparkline
background : "gray"
// ...
});
});
}
```



### Stroke color and width

You can customize the series border color and width using **stroke** and **width**. This is applicable for Sparkline types line and area.

#### JAVASCRIPT

```
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
// ...
//Specifies border color and width for line and area series
stroke: "green",
width: 3
// ...
});
});
```



### Sparkline border

You can customize the **border** width and height of the Sparkline using **border width** and **border height** properties. This is applicable for column, win-loss and pie series.

#### JAVASCRIPT

```
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
//Specifies border width and color for column, winLoss and pie series
border: { color: "green", width: 2 },
// ...
});
});
```



### Opacity

By default **opacity** of the Sparkline is 1. You can specify the opacity value from 0 to 1. This is applicable for all types of series.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
    //..  
    //Specifies the opacity of the sparkline  
    opacity: 0.5  
    // ...  
  });  
});
```



### Localization

Sparkline is having support for localization as well. Default culture is "en-US". You can modify the culture using the property **locale**.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
    // ...  
    //Culture for the sparkline  
    culture: "fr-FR"  
    // ...  
  });  
});
```

### Padding for Sparkline

**Padding** is used to specify the padding value between the container and Sparkline. By default padding value of the Sparkline is 5.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
    // ...  
    //Padding for the sparkline  
    padding: 20,  
    // ...  
  });  
});
```



### Canvas support

You can control whether Sparkline has to be rendered as SVG or **Canvas**. Canvas rendering supports all the functionalities supported in SVG rendering.

#### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
    // ...  
    //enables canvas rendering  
    enableCanvasRendering : true  
    // ...  
  });  
});
```

## Themes

You can specify different **themes** for Sparkline control.

### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
    // ...  
    //theme for sparkline  
    theme: "flatdark",  
    // ...  
  });  
});
```



## Methods

### redraw()

Redraws the entire sparkline. You can call this method whenever you update, add or remove points from the data source or whenever you want to refresh the UI.

### HTML

```
<div id="sparkline">Sparkline</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module SparklineComponent {  
  $(function () {  
    var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
      ///...  
    });  
    // Redraws the Sparkline  
    sample.redraw();  
  });  
};
```

## Events

### load

Fires before loading the sparkline.

### JAVASCRIPT

```
<script>
//load event for sparkline
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
load: function () {
//...//
}
});
});
</script>
```

#### *loaded*

Fires after loaded the sparkline.

#### **JAVASCRIPT**

```
<script>
//loaded event for sparkline
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
loaded: function () {
//...//
}
});
});
</script>
```

#### *tooltipInitialize*

Fires before rendering trackball tooltip. You can use this event to customize the text displayed in trackball tooltip.

#### **JAVASCRIPT**

```
<script>
//tooltip initialize event for sparkline
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
tooltipInitialize: function () {
//...//
}
});
});
</script>
```

#### *seriesRendering*

Fires before rendering a series. This event is fired for each series in Sparkline.

#### **JAVASCRIPT**

```
<script>
//seriesRendering event for sparkline
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
seriesRendering: function () {
//...//
}
});
});
```

```
}  
});  
});  
</script>
```

#### *pointRegionMouseMove*

Fires when mouse is moved over a point.

#### **JAVASCRIPT**

```
<script>  
//pointRegionMouseMove event for sparkline  
$(function () {  
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
pointRegionMouseMove: function () {  
//...//  
}  
});  
});  
</script>
```

#### *pointRegionMouseClick*

Fires on clicking a point in sparkline. You can use this event to handle clicks made on points.

#### **JAVASCRIPT**

```
<script>  
//pointRegionMouseClick event for sparkline  
$(function () {  
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
pointRegionMouseClick: function () {  
//...//  
}  
});  
});  
</script>
```

#### *sparklineMouseMove*

Fires on moving mouse over the sparkline.

#### **JAVASCRIPT**

```
<script>  
//mouse move event for sparkline  
$(function () {  
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {  
sparklineMouseMove: function () {  
//...//  
}  
});  
});  
</script>
```

### *sparklineMouseLeave*

Fires on moving mouse outside the sparkline.

#### **JAVASCRIPT**

```
<script>
//mouse leave event for sparkline
$(function () {
var sample = new ej.datavisualization.Sparkline($("#Sparkline"), {
sparklineMouseLeave: function () {
//...//
}
});
});
});
</script>
```

## SplitButton

### Overview

The **SplitButton** allows you to perform an action by clicking the button and choosing extra options from the dropdown button. The **Split Button** can also display both text and images. The drop down button option is given to the right most corner of the button.

#### **Key Features**

- **Trendy Look** : Rich appearance with Theme Support
- **RTL** : Supports for right to left alignment
- **Text and Image** : Supports both text and image as **SplitButton** content
- **Built-in Icon** : Supports the built-in icon libraries
- **Easy Customization** : The customization of **SplitButton** control to any form is made simple

### Create a simple SplitButton in TypeScript

You can create a **TypeScript** application with the help of the given

<https://help.syncfusion.com/js/typescript>.

Create an **HTML** page and add the scripts references in the order, mentioned in the above link Create the **SplitButton** control as follows.

#### **HTML**

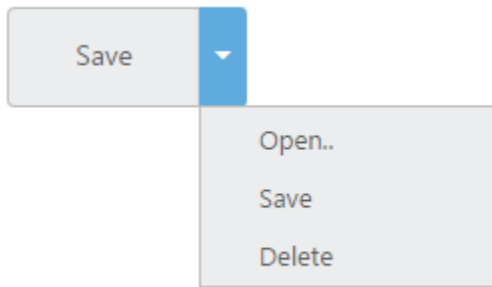
```
<body>
<button id="splitbutton_normal"></button>
<ul id="menu1">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</body>
```

#### **JAVASCRIPT**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```
module ButtonComponent {
  $(function () {
    var splitbutton_normal = new ej.SplitButton($("#splitbutton_normal"), {
      targetID: "menu1"
    });
  });
}
```

You can execute the above code example to display the **SplitButton** control.



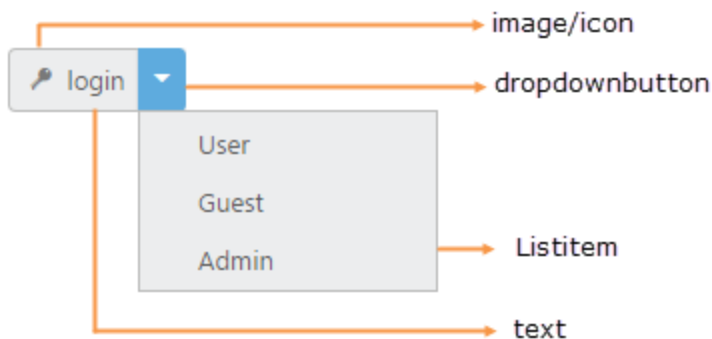
### Configuring SplitButton Properties

This section encompasses the details on how you can configure the SplitButton control in your application and customize it with various properties such as various size, resizing the **SplitButton** according to your requirement.

To render the SplitButton with required text, size and with rounded corner add the following code example in your TS file.

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ButtonComponent {
  $(function () {
    var splitbuttonnormal = new ej.SplitButton($("#splitbuttonnormal"), {
      showRoundedCorner: true,
      size: "large",
      prefixIcon: "e-icon e-key",
      targetID: "menu1",
      contentType: "textandimage",
      text: "login"
    });
  });
}
```

The following screenshot illustrates the **Button** control with text, size and rounded corner properties.



### Easy customization

**Split Button** is used in many applications. **Split Button** Size, Content and content location is varied according to each application. The following sections describes you some customizable options for **Split Button** that can perform easily.

#### Content for Split button

The **targetID** is a mandatory one, without this field it acts as normal button on two sides. This **targetID** property is used to specify the list of content for **Split Button**. The list of content is rendered as a vertical menu list. This vertical menu list is open, when you click on the down arrow of the **Split Button**.

The following steps explains you the details about rendering the **Split Button** with content.

In the **HTML** page, add the following button elements to configure **Split Button** widget.

#### HTML

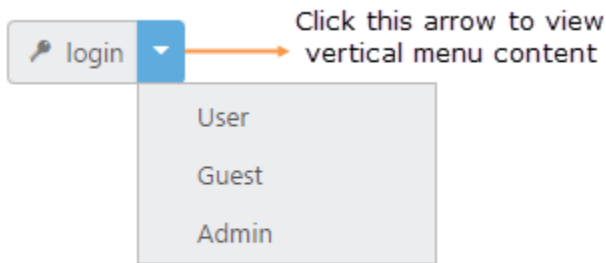
```
<div class="spltsan">
<button id="splitButton">login</button>
<ul id="U111">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitButtonComponent {
$(function () {
var button = new ej.SplitButton($("#splitButton"), {
size: "small",
contentType: "textandimage",
showRoundedCorner: true,
//targetID used to specify vertical menu content
targetID: "U111",
prefixIcon: "e-icon e-login"
});
});
}
```



Execute the above code to render the following output.



### Button Size

You can render the **Split Button** in different sizes. The following table contains some predefined size option for rendering a **Split Button** in easiest way. Each size option has different height and width. Mainly it avoids the complexity in rendering **Split Button** with complex **CSS** class.

List of Button size

| Button Size | Description   |
|-------------|---|
| normal      | Creates split button with content size.                                 |
| mini        | Creates split button with Built-in mini size height, width specified.   |
| small       | Creates split button with Built-in small size height, width specified.  |
| medium      | Creates split button with Built-in medium size height, width specified. |
| Large       | Creates split button with Built-in large size height, width specified.  |

Apart from the above mentioned predefined size option, you can set your own width and height for **Split Button** using **height** and **width** property.

The following steps explains you the details about rendering the **Split Button** with above mentioned size options.

In the **HTML** page, add the following button elements to configure **Split Button** widget.

### HTML

```
<div class="align">
<table>
<tr>
<td class="btnsht">
<div class="spltspan">
<button id="splitButton_normal">login</button>
<ul id="U111">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
```

```

</td>
<td>
<button id="splitButton_mini">login</button>
<ul id="U121">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
<button id="splitButton_small">login</button>
<ul id="U131">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
<button id="splitButton_medium">login</button>
<ul id="U141">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
<button id="splitButton_large">login</button>
<ul id="U151">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
<button id="splitButton_customSize">login</button>
<ul id="U161">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
</tr>
</table>
</div>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitButtonComponent {
$(function () {
var button = new ej.SplitButton($("#splitButton_normal"), {
//Here we used size property to render the split button in different
sizes
//normal size type is used
size: "normal",

```

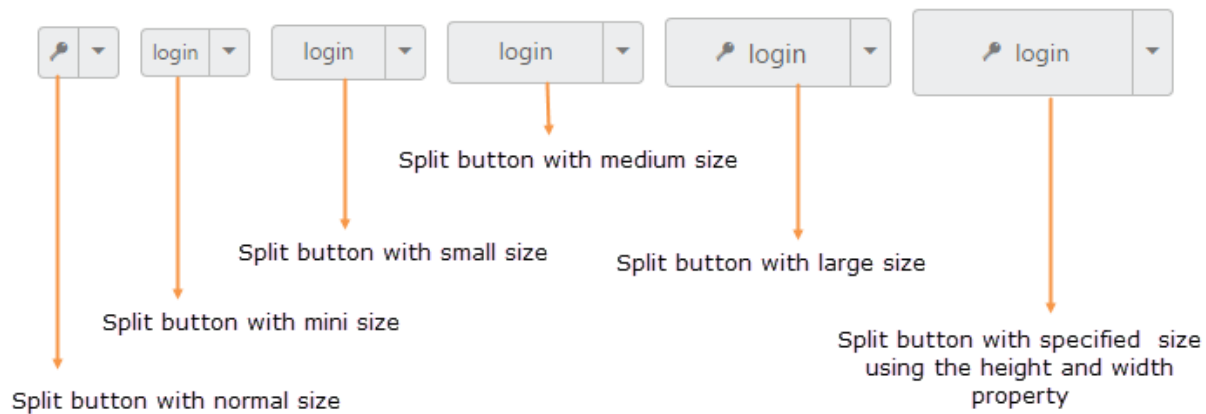
```
showRoundedCorner: true,
contentType: "imageonly",
targetID: "U111",
prefixIcon: "e-icon e-login"
});
var button1 = new ej.SplitButton($("#splitButton_mini"), {
//mini size type is used
size: "mini",
showRoundedCorner: true,
targetID: "U121",
});
var button2 = new ej.SplitButton($("#splitButton_small"), {
//small size type is used
size: "small",
showRoundedCorner: true,
targetID: "U131",
});
var button3 = new ej.SplitButton($("#splitButton_medium"), {
//medium size type is used
size: "medium",
showRoundedCorner: true,
targetID: "U141",
});
var button4 = new ej.SplitButton($("#splitButton_large"), {
//large size type is used
size: "large",
showRoundedCorner: true,
targetID: "U151",
contentType: "textandimage",
prefixIcon: "e-login e-icon",
});
var button5 = new ej.SplitButton($("#splitButton_customSize"), {
//user given height and width
height: 50,
width: 150,
showRoundedCorner: true,
targetID: "U161",
contentType: "textandimage",
prefixIcon: "e-login e-icon",
});
});
}
```

Configure the styles.

### CSS

```
<style>
.control {
width: 500px;
}
.e-split {
float: left;
padding-left: 10px;
}
</style>
```

Execute the above code to render the following output.



### Content Type

The content of the **Split Button** is mainly rendered as text and images. Instead of using complex **CSS** classes to render **Split Button** with different content types, you can use some predefined content type options listed in the following table. Using this content types you can easily add different types of content for **Split Button**. Split Button supports the following content types.

List of content types

| Content Type   | Description                                       |
|----------------|---|
| textonly       | Supports only for text content only.              |
| imageonly      | Supports only for image content only              |
| imageboth      | Supports image for both ends of the button.       |
| textandimage   | Supports image with the text content.             |
| imagetextimage | Supports image with both ends and middle in text. |

### Prefix and Suffix icons

Icons inside the **Split Button** is added easily using **prefixIcon** and **suffixIcon** property. Location of the icon in **Split Button** is a necessary thing. You can customize the location of Icon easily using the following mentioned options.

**Split Button** control also supports the Built-in icon libraries. The **ej.widgets.core.min** CSS contains definitions for important icons that are used in **Split Buttons**. You can use these Built-in icons by mentioning the icon class name as value in **prefixIcon** and **suffixIcon** property. You can use any font icons that is defined in **ej.widgets.core.min** CSS. It avoids the complexity in specifying icon using sprite image and CSS.

For example the following Built-in CSS class are used to display the font icons that is used by media player.

- e-mediaback
- e-mediaforward
- e-medianext
- e-mediaprev
- e-mediaeject
- e-mediaclose
- e-mediapause
- e-mediaplay

### Prefix Icon

It inserts the icon at the starting position of **Split Button**. After this prefix icon, you can use text or suffix icon.

### Suffix Icon

It inserts the icon at the ending position of **Split Button**. Before this suffix icon, you can use text or prefix icon.

The following steps explain you the details on rendering the **Split Button** with above mentioned **content type**, **prefix** and **suffix icon** options

In the **HTML** page, add the following button elements to configure **Split Button** widget.

### HTML

```
<table>
<tr>
<td class="btnsht">
<button id="splitButton_imageonly">login</button>
<ul id="U111">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td>
<button id="splitButton_textonly">login</button>
<ul id="U121">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
<button id="splitButton_imageboth">login</button>
<ul id="U131">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
<button id="splitButton_textandimage">login</button>
<ul id="U141">
<li><span>User</span></li>
```

```

<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
<button id="splitButton_imagegettextimage">login</button>
<ul id="U151">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
</tr>
</table>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitButtonComponent {
$(function () {
var Button = new ej.SplitButton($("#splitButton_imageonly"), {
size: "medium",
//only image is used as content
contentType: "imageonly",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
targetID: "U111",
});
var Button1 = new ej.SplitButton($("#splitButton_textonly"), {
size: "medium",
//only text is used as content
contentType: "textonly",
showRoundedCorner: true,
targetID: "U121",
});
var Button2 = new ej.SplitButton($("#splitButton_imageboth"), {
size: "medium",
//only images in both end is used as content
contentType: "imageboth",
showRoundedCorner: true,
//e-handup is a built-in class, which specify font icon
prefixIcon: "e-icon e-handup",
//e-palette is a built-in class, which specify font icon
suffixIcon: "e-icon e-palette",
targetID: "U131",
});
var Button3 = new ej.SplitButton($("#splitButton_textandimage"), {
size: "medium",
//text and image is used as content
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
targetID: "U141",
});
var Button4 = new ej.SplitButton($("#splitButton_imagegettextimage"), {

```

```

size: "medium",
//images in both end and text in center is used as content
contentType: "imagetextimage",
showRoundedCorner: true,
//It specifies the image in prefix location
prefixIcon: "e-icon e-handup",
//It specifies the image in suffix location
suffixIcon: "e-icon e-palette",
targetID: "U151",
});
});
}

```

Configure the styles.

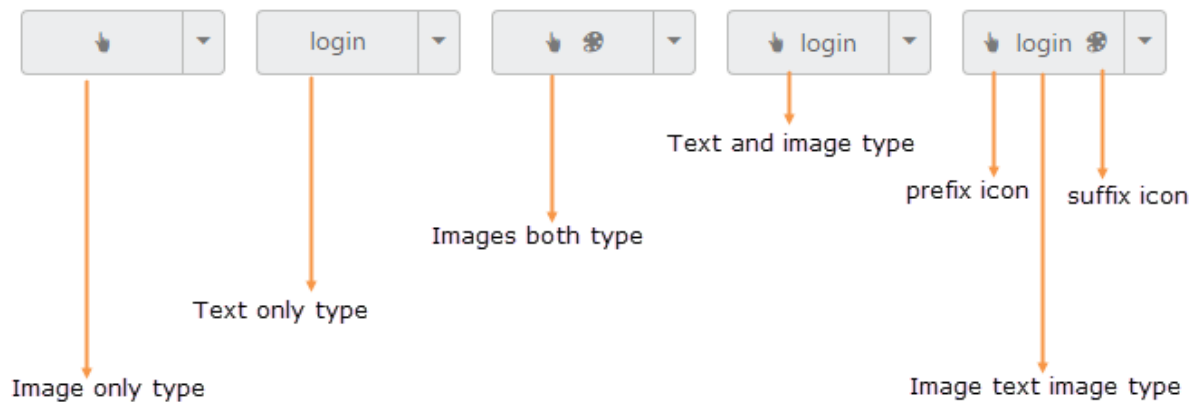
### CSS

```

<style>
.control {
width: 500px;
}
.e-split {
float: left;
padding-left: 15px;
}
</style>

```

Execute the above code to render the following output.



### Image Position

To provide the best look and feel for **Split Button**, position of images in **Split Button** is important. Using **imagePosition** property, you can easily customize the position of images inside **Split Button** without using any complex CSS. **imagePosition** property is applicable only with the **textandimage** content type property. This property represent the position of images with respect to text.

### Property Table

| Image Position | Description |
|----------------|-------------|
|----------------|-------------|

|             |   |
|-------------|---|
| imageleft   | Support for aligning text in right and image in left. |
| imageright  | Support for aligning text in left and image in right. |
| imagetop    | Support for aligning text in bottom and image in top. |
| imagebottom | Support for aligning text in top and image in bottom. |

The following steps explains you the details on rendering the **Split Button** with above mentioned image position options.

In the **HTML** page, add the following button elements to configure **Split Button** widget.

### HTML

```
<table>
<tr>
<td class="btnsht">
<button id="splitButton_imageleft_normal">login</button>
<ul id="U111">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td>
<button id="splitButton_imageleft_small">login</button>
<ul id="U121">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
<button id="splitButton_imageleft_medium">login</button>
<ul id="U131">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
<button id="splitButton_imageleft_large">login</button>
<ul id="U141">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
</tr>
<tr>
<td class="btnsht">
<button id="splitButton_imageright_normal">login</button>
<ul id="U11">
<li><span>User</span></li>
<li><span>Guest</span></li>
```



```

<li><span>Admin</span></li>
</ul>
</td>
<td>
<button id="splitButton_imageright_small">login</button>
<ul id="U12">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
<button id="splitButton_imageright_medium">login</button>
<ul id="U13">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
<button id="splitButton_imageright_large">login</button>
<ul id="U14">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
</tr>
<tr>
<td class="btnsht">
<button id="splitButton_imagetop">login</button>
<ul id="U15">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td>
<button id="splitButton_imagebottom">login</button>
<ul id="U16">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
</tr>
</table>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitButtonComponent {
$(function () {
var Button = new ej.SplitButton($("#splitButton_imageleft_normal"), {
imagePosition: "imageleft",

```

```
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
targetID: "U11",
});
var Button1 = new ej.SplitButton($("#splitButton_imageonly"), {
size: "small",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
targetID: "U121",
});
var Button2 = new ej.SplitButton($("#splitButton_imageonly"), {
size: "medium",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
targetID: "U131",
});
var Button3 = new ej.SplitButton($("#splitButton_imageonly"), {
size: "large",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
targetID: "U11",
});
var Button4 = new ej.SplitButton($("#splitButton_imageonly"), {
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
targetID: "U12",
});
var Button5 = new ej.SplitButton($("#splitButton_imageonly"), {
size: "small",
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
targetID: "U13",
});
var Button6 = new ej.SplitButton($("#splitButton_imageonly"), {
size: "medium",
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
targetID: "U14",
});
var Button7 = new ej.SplitButton($("#splitButton_imageonly"), {
size: "large",
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
```

```

prefixIcon: "e-icon e-handup",
targetID: "U141",
});
var Button8 = new ej.SplitButton($("#splitButton_imageonly"), {
imagePosition: "imagetop",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
targetID: "U15",
height: 60
});
var Button9 = new ej.SplitButton($("#splitButton_imageonly"), {
imagePosition: "imagebottom",
contentType: "textandimage",
showRoundedCorner: true,
prefixIcon: "e-icon e-handup",
targetID: "U16",
height: 60
});
});
});
}

```

Configure the styles.

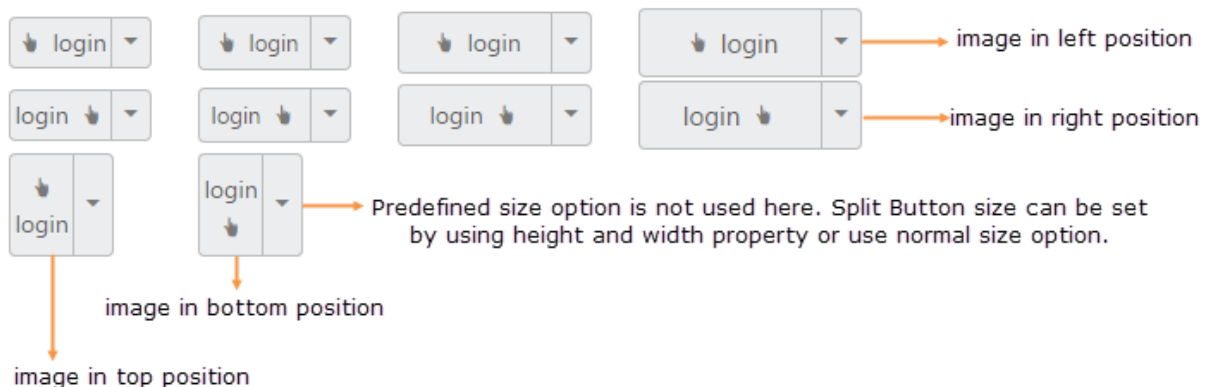
### CSS

```

<style>
.control {
width: 420px;
}
.e-split {
float: left;
padding-left: 25px;
}
</style>

```

Execute the above code to render the following output.



### Theme support

You can control the style and appearance of **Split Button** based on **CSS** classes. In order to apply styles to the **Split Button** control, you can refer two files namely, **ej.widgets.core.min.css** and

**ej.theme.min.css.** When you refer the **ej.widgets.all.min.css** file, then it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.widgets.all.min.css** is the combination of these two.

By default, there are 16 theme support available for RadialMenu component, namely:

- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap
- high-contrast-01
- high-contrast-02
- material
- office-365

### Custom CSS

You can use the **CSS** class to customize the **Split Button** control appearance. Define a **CSS** class as per requirement and assign the class name to **cssClass** property.

The following steps explains you the details about rendering the **Split Button** with custom **CSS**.

In the **HTML** page, add the following button elements to configure **Split Button** widget.

### CSS

```
<div class="align">
<table>
<tr>
<td class="btnsht">
<div class="spltsan">
<button id="splitButton_customCss1">login</button>
<ul id="U111">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
</td>
<td>
<button id="splitButton_customCss2">login</button>
<ul id="U121">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
```

```

</ul>
</td>
<td class="btnsht">
<button id="splitButton_customCss3">login</button>
<ul id="U131">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
<button id="splitButton_customCss4">login</button>
<ul id="U141">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
<button id="splitButton_customCss5">login</button>
<ul id="U151">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
</tr>
</table>
</div>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitButtonComponent {
$(function () {
var Button = new ej.SplitButton($("#splitButton_customCss1"), {
cssClass: "customCss1",
size: "small",
showRoundedCorner: true,
contentType: "textandimage",
prefixIcon: "e-icon e-handup",
targetID: "U111",
});
var Button1 = new ej.SplitButton($("#splitButton_customCss2"), {
cssClass: "customCss2",
size: "small",
showRoundedCorner: true,
contentType: "textandimage",
prefixIcon: "e-icon e-handup",
targetID: "U121",
});
var Button2 = new ej.SplitButton($("#splitButton_customCss3"), {
cssClass: "customCss3",
size: "small",
showRoundedCorner: true,

```

```

contentType: "textandimage",
prefixIcon: "e-icon e-handup",
targetID: "U131",
});
var Button3 = new ej.SplitButton($("#splitButton_customCss4"), {
cssClass: "customCss4",
size: "small",
showRoundedCorner: true,
contentType: "textandimage",
prefixIcon: "e-icon e-handup",
targetID: "U141",
});
var Button4 = new ej.SplitButton($("#splitButton_customCss5"), {
cssClass: "customCss5",
size: "small",
showRoundedCorner: true,
contentType: "textandimage",
prefixIcon: "e-icon e-handup",
targetID: "U151",
});
});
}

```

Configure the **CSS** styles to apply on buttons.

### CSS

```

<style type="text/css">
/* Customize the button background */
.e-split .customCss1 {
background-color: #121111;
}
.e-split .customCss2 {
background-color: #94bbd5;
}
.e-split .customCss3 {
background-color: #f3533c;
}
.e-split .customCss4 {
background-color: #d1eed;
}
.e-split .customCss5 {
background-color: #deb66e;
}
/* Customize the button image & text color */
.e-split .customCss1.e-btn.e-select .e-icon, .e-split .customCss1.e-
btn.e-select .e-btntxt {
color: #94bbd5;
}
.e-split .customCss2.e-btn.e-select .e-icon, .e-split .customCss2.e-
btn.e-select .e-btntxt {
color: #121111;
}
.e-split .customCss3.e-btn.e-select .e-icon, .e-split .customCss3.e-
btn.e-select .e-btntxt {
color: #cef6f7;
}

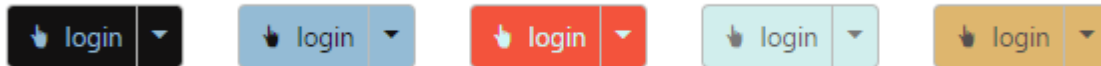
```

```

}
.e-split .customCss5.e-btn.e-select .e-icon, .e-split .customCss5.e-
btn.e-select .e-btntxt {
color: #534f4f;
}
.e-split {
float: left;
padding-left: 25px;
}
</style>

```

Execute the above code to render the following output.



## Dropdown Button

You can change the **Split Button** as **Dropdown Button** that consists of a single button that when clicked displays a drop-down list of mutually exclusive items. You can achieve this by using default functionality of **Split Button** with **buttonMode** as **ej.ButtonMode.Dropdown**. Initially the **targetID** is a mandatory one.

The following steps explain how to change the **Split Button** as **Dropdown Button**.

In the **HTML** page, add the following button elements to configure **Button** widget.

### HTML

```

<button id="dropdownButton">login</button>
<ul id="menu">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>

```

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitButtonComponent {
$(function () {
var button = new ej.SplitButton($("#dropdownButton"), {
size: "medium",
showRoundedCorner: true,
buttonMode: ej.ButtonMode.Dropdown,
targetID: "menu"
});
});
}

```

Execute the above code to render the following output.



### Arrow Position

To provide a good look and feel for **Split Button**, position of arrow in **Split Button** is important. Using **arrowPosition** property, you can easily customize the position of arrow inside **Split Button** without using any complex CSS. **arrowPosition** property is applicable for both **Split Button** and **Dropdown Button**. This property represent the position of arrow with menu content.

List of arrow position

| Arrow Position | Description                  |
|----------------|------------------------------|
| left           | Support for arrow in left.   |
| right          | Support for arrow in right.Â |
| top            | Support for arrow in top.Â   |
| bottom         | Support for arrow in bottom. |

The following steps explain you the details on rendering the **Split Button** with above mentioned arrow position options.

In the **HTML** page, add the following button elements to configure **Split Button** widget.

### HTML

```
<div class="control">
  <button id="spltbody11">login</button>
  <ul id="U11">
    <li><span>User</span></li>
    <li><span>Guest</span></li>
    <li><span>Admin</span></li>
  </ul>
  <button id="spltbody21">login</button>
  <ul id="U12">
    <li><span>User</span></li>
    <li><span>Guest</span></li>
    <li><span>Admin</span></li>
  </ul>
  <button id="spltbody31">login</button>
  <ul id="U13">
    <li><span>User</span></li>
    <li><span>Guest</span></li>
    <li><span>Admin</span></li>
  </ul>
  <button id="spltbody41">login</button>
  <ul id="U14">
```



```
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
```

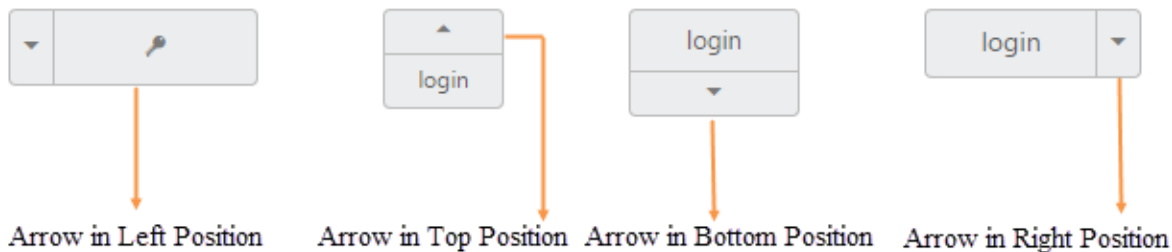
## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitButtonComponent {
$(function () {
var button = new ej.SplitButton($("#spltbody11"), {
size: "large",
showRoundedCorner: true,
contentType: "imageonly",
targetID: "U111",
prefixIcon: "e-icon e-login",
arrowPosition: "left"
});
var button1 = new ej.SplitButton($("#spltbody21"), {
size: "mini",
showRoundedCorner: true,
targetID: "U121",
arrowPosition: "top"
});
var button2 = new ej.SplitButton($("#spltbody31"), {
size: "small",
showRoundedCorner: true,
targetID: "U131",
arrowPosition: "bottom"
});
var button3 = new ej.SplitButton($("#spltbody41"), {
size: "medium",
showRoundedCorner: true,
targetID: "U141",
arrowPosition: "right"
});
});
});
}
```

## CSS

```
<style>
.e-split {
float: left;
padding-left: 65px;
}
</style>
```

Execute the above code to render the following output.



## RTL Support

In some cases it is necessary to use right to left alignment. **RTL** support is provided by using **enableRTL** property. In **RTL** mode when there is more than one content (image/text, image/image) in **Split Button**, then the content is aligned in right to left format. For example, when text is in left and image is in right position, after applying right to left alignment these position are interchanged.

The following steps explain you the details about rendering the button with Right to left alignment support

In the **HTML** page, add the following button elements to configure **Split Button** widget.

### HTML

```
<div class="spltspan">
  <button id="splitButton_rtl">login</button>
  <ul id="U111">
    <li><span>User</span></li>
    <li><span>Guest</span></li>
    <li><span>Admin</span></li>
  </ul>
</div>
```

### JAVASCRIPT

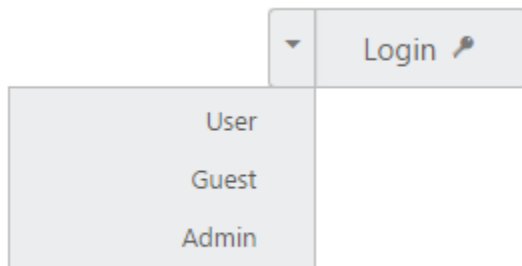
```
// Initialize the control in JavaScript
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitButtonComponent {
  $(function () {
    var Button = new ej.SplitButton($("#splitButton_rtl"), {
      //used to enable or disable RTL support for split button
      enableRTL: true,
      size: "large",
      contentType: "textandimage",
      showRoundedCorner: true,
      targetID: "U111",
      prefixIcon: "e-icon e-login"
    });
  });
}
```

Configure the styles

### CSS

```
<style>
.spltspan {
margin-left: 120px;
}
</style>
```

Execute the above code to render the following output.



## Miscellaneous

### Text

It is necessary to display the user defined text for **Split Button**. Using **text** property, you can easily set text content for **Split Button**. This text property overwrites the text that is provided on input button element.

The following steps explain you the details about rendering the **Split Button** with specified text.

In the **HTML** page, add the following button elements to configure **Split Button** widget.

### HTML

```
<div class="spltspan">
<button id="splitButton_text">login</button>
<ul id="U111">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
```

### JAVASCRIPT

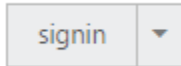
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitButtonComponent {
$(function () {
var Button = new ej.SplitButton($("#splitButton_text"), {
//used to set the text content for split button
text: "signIn",
size: "small",
targetID: "U111"
});
});
}
```

Configure the styles.

### CSS

```
<style>
.spltspan {
margin-left: 120px;
}
</style>
```

Execute the above code to render the following output.



In output “login” text in **Split Button** is replaced by text property value.

### Show Rounded Corner

Specifies the corner of **Split Button** in rounded shape. By default, the edges of **Split Button** is not rounded. To set rounded corner, you can enable **showRoundedCorner** property.

The following steps explains you the details about rendering the **Split Button** with rounded corner.

In the **HTML** page, add the following button elements to configure **Split Button** widget.

### HTML

```
<div class="spltspan">
<button id="splitButton_roundedCorner">login</button>
<ul id="U111">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitButtonComponent {
$(function () {
var Button = new ej.SplitButton($("#splitButton_roundedCorner"), {
size: "small",
//Enable or disable the rounded corner for split button
showRoundedCorner: true,
targetID: "U111"
});
});
}
```

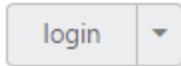
Configure the styles.

### CSS

```
<style>
```

```
.spltspan {  
margin-left: 120px;  
}  
</style>
```

Execute the above code to render the following output.



## Splitter

### Overview

The Typescript Splitter control that enables you to divide a Web page into distinct areas by inserting resizable panes. You can create any number of Splitter panes and place them inside the Splitter control. The split bars are inserted automatically in between the adjacent panes.

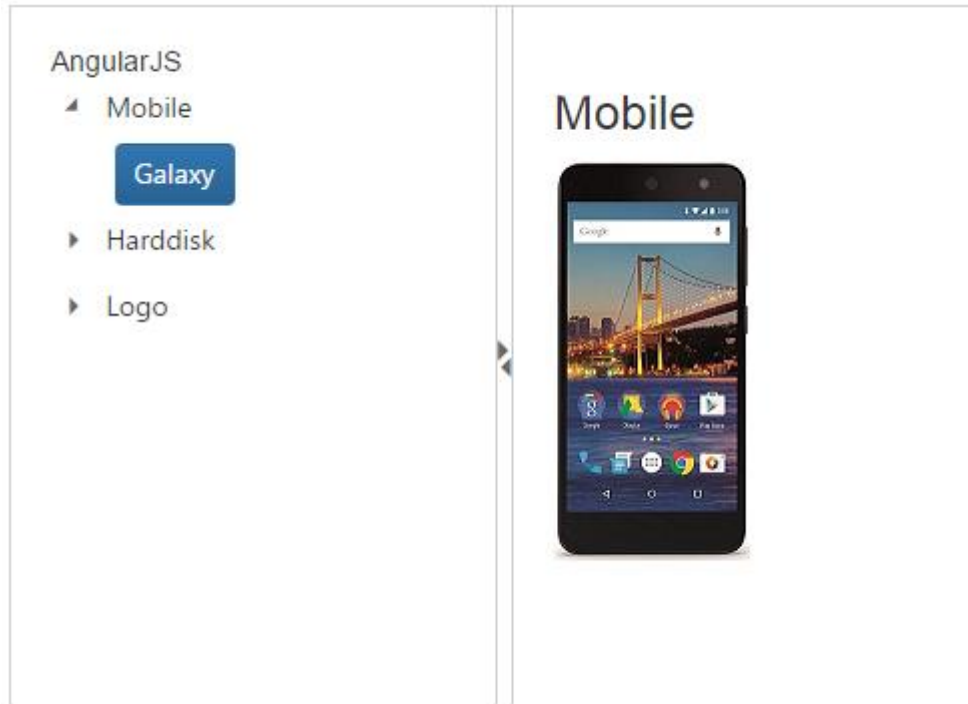
### Key Features

- **Expand or Collapse Panes:** Support for expanding or collapsing panes by using the Expand or Collapse arrows.
- **Animation:** Supports for animating panes while expanding or collapsing the panes, and also setting the animation speed.
- **Window Resizing:** Supports for resizing the Splitter while resizing the window.
- **RTL:** Supports for right to left alignment of Splitter panes.

### Getting Started

**Splitter** control consists of movable split bar(s) that divides a container's display area into two or more resizable and collapsible panels.

From the following guidelines, you can create a Splitter, add Tree view in the Splitter and set actions to view the image. It is used to split the document or image and Expand or Collapse in the Splitter. The following screenshot demonstrates the functionality of Splitter control.



### Create a Splitter

Refer the common Typescript [Getting Started Documentation](#) to create an application and add necessary scripts and styles for rendering the Splitter control.

Create an HTML file and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Typescript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/fl
at-azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.
web.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add Splitter here-->
</body>
</html>
```

Create div element and add in the body tag. To create the Splitter, you should call the `ejSplitter` jQuery plug-in function.

#### HTML

```
<div id="outterSplitter">
```

```
</div>
```

Initialize the Splitter in app.ts file by using the ej.Splitter method.

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module SplitterComponent {
$(function () {
let sample = new ej.Splitter($("#outterSplitter"));
});
}
```

Now build your application, so that the app.js file is automatically generated and got added to your project (User have nothing to do with this file). Now, whatever code changes that you make in app.ts file will be reflected in app.js file automatically.

### Configure Splitter Panes

Add div element to create Splitter. Save the images in the corresponding location.

### HTML

```
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area" style="height:400px; margin:0 auto;">
<div id="outterSplitter">
<div>
<div class="cont">
<h3 class="h3">Typescript</h3>
</div>
</div>
<div>
<div class="cont">
<div class="_content">
Select any product from the tree to show the description.
</div>
<div class=" mobile spe" style="display:none">
<h3>Mobile</h3>

</div>
<div class=" harddisk spe" style="display:none">
<h3>Harddisk</h3>

</div>
<div class="logo spe" style="display:none">
<h3>Logo</h3>

</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
```

Add the following styles to show the Splitter control in horizontal order.

### CSS

```
.cont {  
padding: 10px 0 0 10px;  
text-align: center;  
}
```

### Configure Tree View

For adding Tree View control, you have to use **ul** element to corresponding element.

Add the following code example in HTML file to configure Tree View.

### HTML

```
<ul id="treeView" class="visibleHide">  
<li> Mobile  
<ul>  
<li id=" Galaxy " class="_child">Galaxy</li>  
</ul>  
</li>  
<li>Harddisk  
<ul>  
<li id="Harddisk" class="_child">Segate </li>  
</ul>  
</li>  
<li>Logo  
<ul>  
<li id="Logo" class="_child">Amazon</li>  
</ul>  
</li>  
</ul>
```

### Set Actions

Add the following code example in the typescript to set the action to view the image. We need to create instance of TreeView to achieve **nodeSelect** event of TreeView.

### JAVASCRIPT

```
module SplitterComponent {  
$(function () {  
var splitterInstance = new ej.Splitter($("#outterSplitter"), {  
height: "250px",  
width: "50%",  
properties: [{paneSize:"50%"}, { }],  
isResponsive:true  
});  
var treeViewInstance = new ej.TreeView($("#treeView"), {  
nodeSelect: (e) => {  
if (e.currentElement.hasClass('e-item last')) {  
var content = $('.' + e.currentElement[0].id).html();  
$('._content').html(content);  
}  
},  
},
```

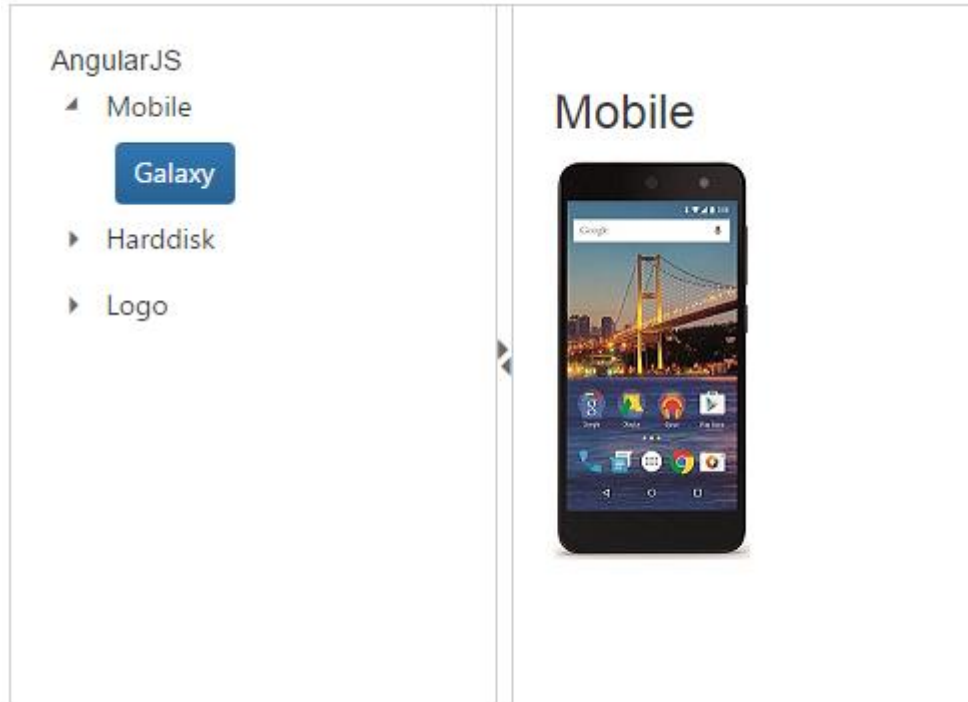


```

    });
  });
}

```

The following screenshot is the output for the above code.



*Note:-> You can find the Splitter properties from the [API reference](#) document*

### Configure Collapsible and Expandable properties

The **Splitter** provides you the option to enable or disable the pane collapse functionality. You can click the icon in **Split bar** to collapse or expand the corresponding pane element in **Splitter**. Setting the **collapsible** property to “false” disables the pane collapse functionality in the **Splitter** widget.

### Enabling Collapsible

The following steps explain the implementation of the **collapsible** option in **Splitter**.

In the **HTML** page set the **<div>** element to render **Splitter** control.

### HTML

```

<div id="splitter">
  <div>
    <div style="padding: 0px 15px;">
      <h3 class="h3">Tools </h3>
      Essential Tools is an collection of user interface components used to
      create interactive
      JavaScript applications.
    </div>
    </div>
    <div>
      <div style="padding: 0px 15px;">
        <h3 class="h3">Grid </h3>

```

Essential JavaScript Grid offers full featured a Grid control with extensive support for Grouping and the display of hierarchical data.

</div>

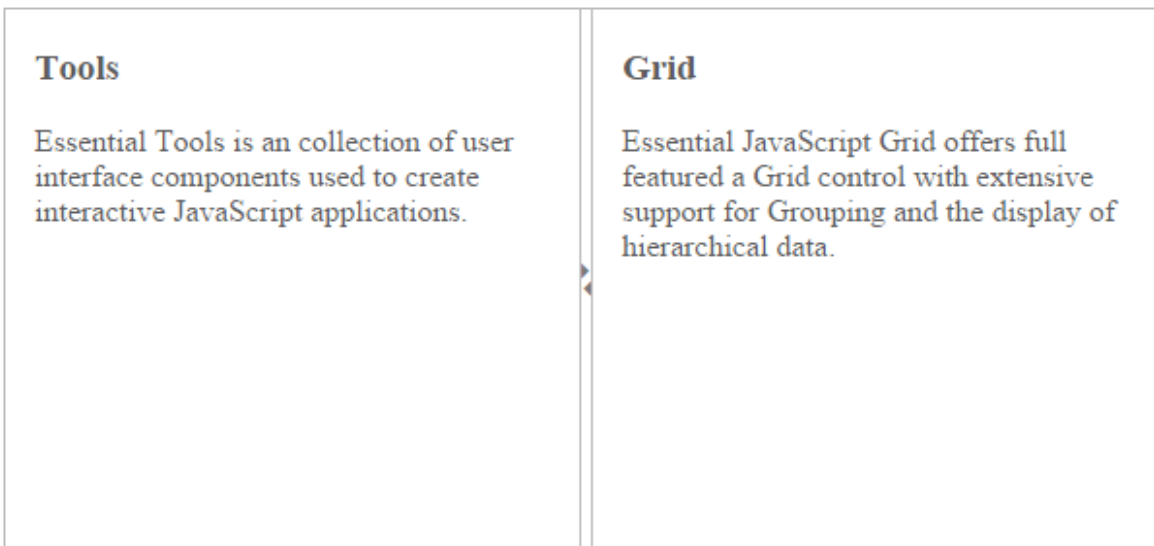
</div>

</div>

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitterComponent {
  $(function () {
    var splitterInstance = new ej.Splitter($("#splitter"), {
      height: 280, width: 600,
      properties: [{collapsible: true}]
    });
  });
}
```

The output for **Splitter** when **collapsible** is set to “True” is as follows.



The output for Splitter when **collapsible** is “false”.

|   |  |
|---|--|
| <h3>Tools</h3> <p>Essential Tools is an collection of user interface components used to create interactive JavaScript applications.</p> | <h3>Grid</h3> <p>Essential JavaScript Grid offers full featured a Grid control with extensive support for Grouping and the display of hierarchical data.</p> |
|---|--|

#### Disable Expandable

The following steps explain the implementation of the **expandable** option in **Splitter**.

In the **HTML** page set the **<div>** element to render **Splitter** control.

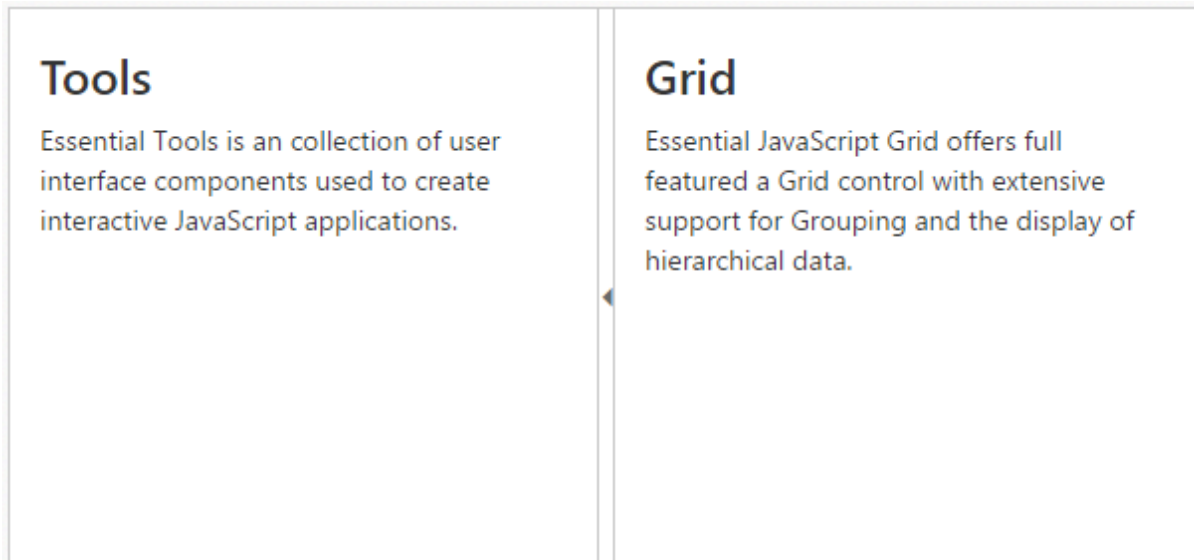
#### HTML

```
<div id="splitter">
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Tools </h3>
Essential Tools is an collection of user interface components used to
create interactive
JavaScript applications.
</div>
</div>
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Grid </h3>
Essential JavaScript Grid offers full featured a Grid control with
extensive support for
Grouping and the display of hierarchical data.
</div>
</div>
</div>
```

#### JAVASCRIPT

```
var splitterInstance = new ej.Splitter($("#splitter"), {
height: 280, width: 600,
properties: [{expandable:false},{collapsible: false}]
});
```

The output for Splitter when **expandable** is "false".



### Splitter Orientation

The **Splitter** supports both vertical and horizontal orientation of the pane. You can declare the orientation by using **enum**, **ej.Orientation.Vertical** or **ej.Orientation.Horizontal**, that have the corresponding value of vertical and horizontal as a string.

### Configure Splitter Orientation

The following steps explain the implementation of **Splitter** orientation option.

In the **HTML** page set the **<div>** element for rendering **Splitter** widget.

#### HTML

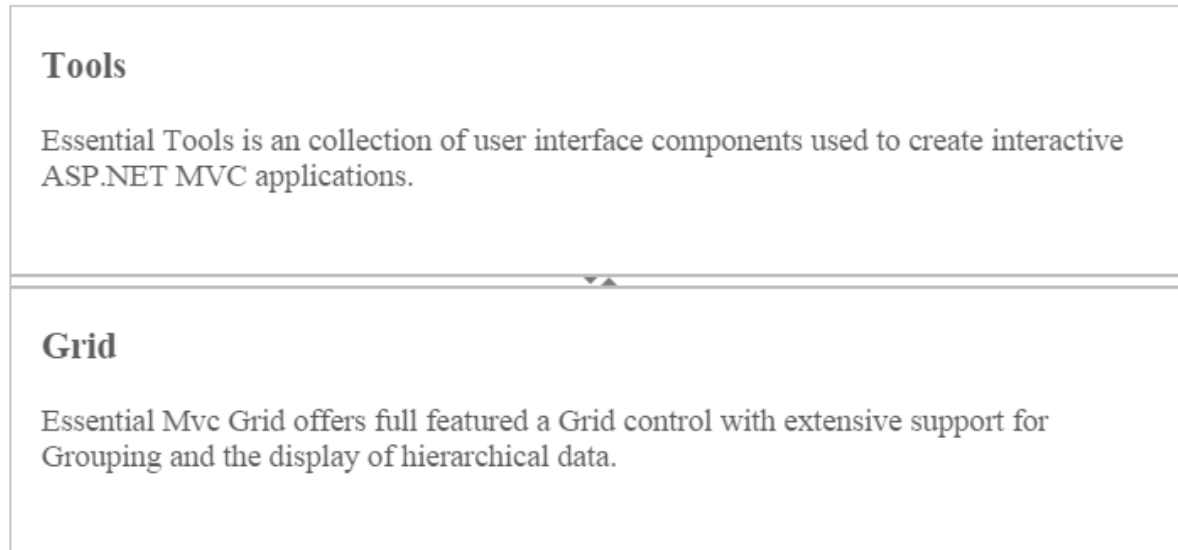
```
<div id="splitter">
  <div>
    <div style="padding: 0px 15px;">
      <h3 class="h3">Tools </h3>
      Essential Tools is an collection of user interface components used to
      create interactive
      ASP.NET MVC applications.
    </div>
  </div>
  <div>
    <div style="padding: 0px 15px;">
      <h3 class="h3">Grid </h3>
      Essential MVC Grid offers full featured a Grid control with extensive
      support for
      Grouping and the display of hierarchical data.
    </div>
  </div>
</div>
```

#### JAVASCRIPT

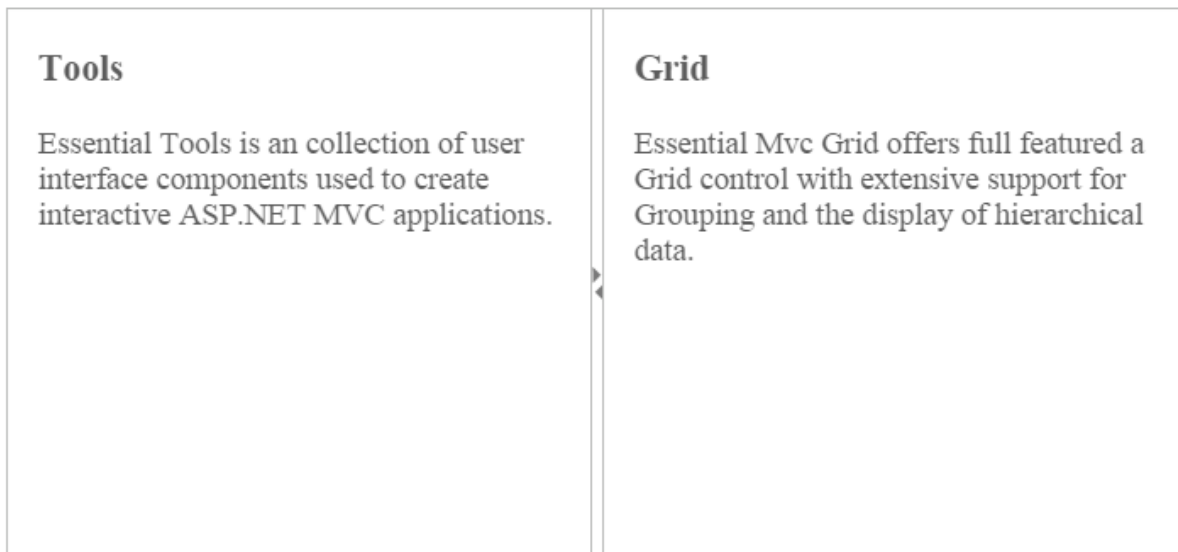
```
module SplitterComponent {
  $(function () {
    var splitterInstance = new ej.Splitter($("#splitter"), {
      height: 280, width: 600,
```

```
orientation: ej.Orientation.Vertical,
});
});
}
```

The output for **Splitter** with **ej.Orientation.Vertical**.



The output for **Splitter** with **ej.Orientation.Horizontal**.



## Nested Splitter Support

The **Splitter** provides nested pane support that allows you to add a pane between two pane elements.

### Configure Nested Splitter

The following steps explain the implementation of the “**nested splitter**” option.

In the **HTML** page set the corresponding **<div>** elements for outer and inner **Splitter** control.

### HTML

```

<div id="outersplitter">
  <div>
    <div style="padding: 0px 15px;">
      <h3 class="h3"> JavaScript </h3>
    </div>
  </div>
  <div id="innersplitter">
    <div>
      <div style="padding: 0px 15px;">
        <h3 class="h3">Tools </h3>
        Essential Tools is an collection of user interface components used to
        create interactive
        JavaScript applications.
      </div>
    </div>
    <div>
      <div style="padding: 0px 15px;">
        <h3 class="h3">Chart </h3>
        Essential Chart is a business-oriented charting component.
      </div>
    </div>
    <div>
      <div style="padding: 0px 15px;">
        <h3 class="h3">Grid </h3>
        Essential JavaScript Grid offers full featured a Grid control with
        extensive support for
        Grouping and the display of hierarchical data.
      </div>
    </div>
  </div>
</div>

```

## JAVASCRIPT

```

module SplitterComponent {
  $(function () {
    var splitterInstance = new ej.Splitter($("#outterSpliter"), {
      height: 280, width: 600,
      orientation: ej.Orientation.Vertical,
      properties: [{ paneSize: 60 }]
    });
    var splitterInstance1 = new ej.Splitter($("#innerSpliter"), {
      width: 600,
      properties: [{ paneSize: 200 }, { paneSize: 170 }]
    });
  });
}

```

The output for **nested Splitter**.

| JavaScript  |  |  |
|---|--|--|
| <b>Tools</b><br><br>Essential Tools is an collection of user interface components used to create interactive JavaScript applications. | <b>Chart</b><br><br>Essential Chart is a business-oriented charting component. | <b>Grid</b><br><br>Essential JavaScript Grid offers full featured a Grid control with extensive support for Grouping and the display of hierarchical data. |

## Splitter Integration

The **Splitter** allows you to use other **Essential JavaScript** products inside the pane. The integrated function of those widgets can be used in other panes of the **Splitter**.

### Configuring other widgets in Splitter

The following steps explain the implementation of **Splitter** integration.

In the **HTML** page set the **<div>** element for rendering **Splitter** with two panes. The first pane has the **TreeView** content and the next one has some content that is related to **TreeView**.

### HTML

```
<div id="splitter">
<div>
<div style="padding: 20px;">
<h3> JavaScript </h3>
<ul id="treeview">
<li>
Tools
<ul>
<li id="tools" class="_child">Description</li>
</ul>
</li>
<li>
Chart
<ul>
<li id="chart" class="_child">Description </li>
</ul>
</li>
<li>
Grid
<ul>
<li id="grid" class="_child">Description</li>
</ul>
</li>
</ul>
</div>
```

```

</div>
<div>
<div style="padding: 20px">
<div class="_content">
Select any product from the tree to show the description.
</div>
<div class="tools" style="display: none">
<h3>Tools</h3>
Essential Tools is an collection of user interface components used to
create interactive
JavaScript applications.
</div>
<div class="chart" style="display: none">
<h3>Chart</h3>
Essential Chart is a business-oriented charting component.
</div>
<div class="grid" style="display: none">
<h3>Grid</h3>
Essential JavaScript Grid offers full featured a Grid control with
extensive support for
Grouping and the display of hierarchical data.
</div>
</div>
</div>
</div>
</div>

```

## JAVASCRIPT

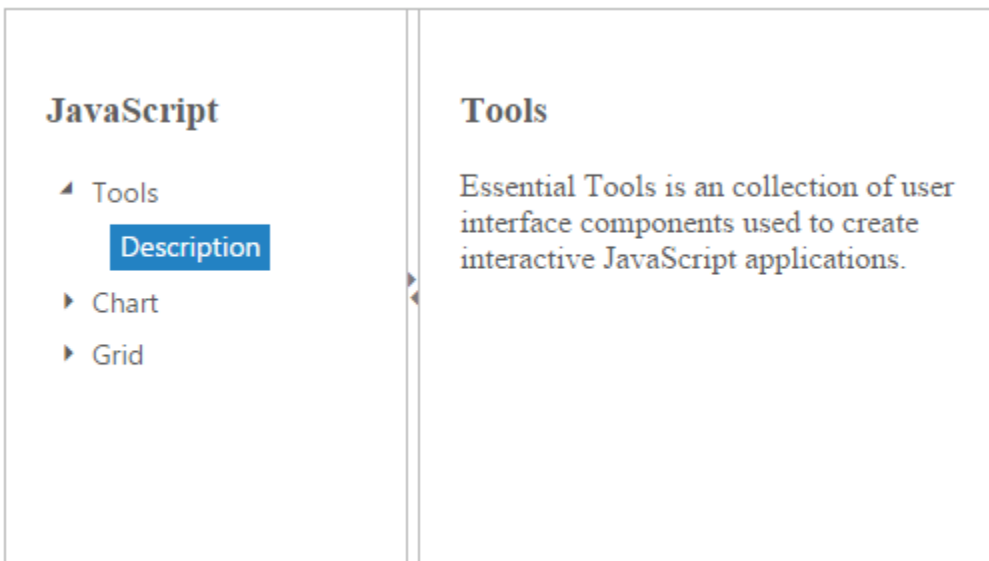
```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitterComponent {
$(function () {
var treeviewInstance = new ej.Splitter($("#treeview"), {
nodeSelect: "treeClicked"
}); /* Render TreeView inside Splitter pane */
var splitter = new ej.Splitter($("#innerSplitter"), {
height: 280, width: 500,
properties: [{ paneSize: 200 }]
});
});
}
function treeClicked(sender, args) { //nodeSelect event handle
if (sender.currentElement.hasClass('_child')) {
var content = $('.' + sender.currentElement[0].id).html();
$('_content').html(content);
}
}
}

```

When the node is selected in **TreeView**, the integrated output is displayed in the second pane.





## Appearance and Styling

### Responsive

The **enableAutoResize** option allows the **Splitter** control to adapt its rendering based on the parent container where it is actually placed. When this option is set to true, the **Splitter** control adjusts its height and width based on the outer container that contains it, and also the sub-elements within it adjust to its height, width and position, appropriately.

### Enabling Auto Resize

The following steps explain the implementation of **AutoResize** option in the **Splitter** control.

In the **HTML** page set the corresponding **<div>** elements for outer and inner splitter control.

### HTML

```
<div id="outersplitter">  
<div>
```

```

<div style="padding: 0px 15px;">
<h3 class="h3">JavaScript </h3>
</div>
</div>
<div id="innersplitter">
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Tools </h3>
Essential Tools is an collection of user interface components used to
create interactive
JavaScript applications.
</div>
</div>
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Grid </h3>
Essential JavaScript Grid offers full featured a Grid control with
extensive support for
Grouping and the display of hierarchical data.
</div>
</div>
</div>
</div>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitterComponent {
$(function () {
var splitterInstance = new ej.Splitter($("#outterSplitter"), {
height: 280, width: "100%",
orientation: ej.Orientation.Vertical,
properties: [{ paneSize: 60 }],
enableAutoResize: true
});
var splitterInstance1 = new ej.Splitter($("#innerSplitter"), {
enableAutoResize: true
});
});
}

```

## Animation Support

The **Splitter** provides you animation support when you expand or collapse the pane. The animation speed can be modified by using the **animationSpeed** property, that has values in milliseconds.

### Enabling Animation with Animation speed

The following steps explain the implementation of **enableAnimation** option in the **Splitter** widget.

In the **HTML** page set the corresponding **<div>** element for rendering **Splitter** control.

## HTML

```

<div id="splitter">
<div>
<div style="padding: 0px 15px;">

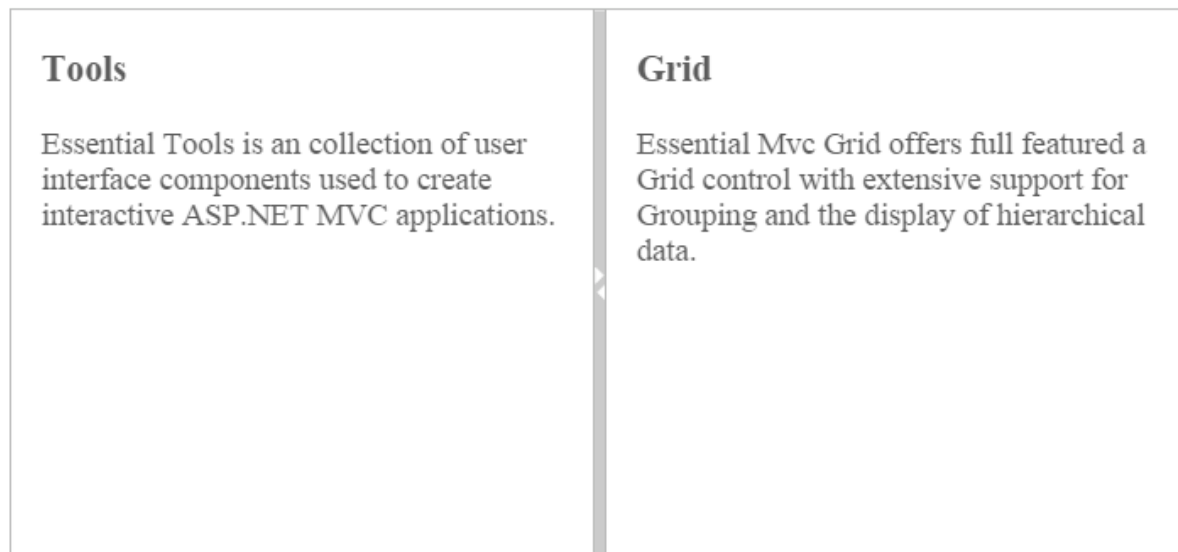
```

```
<h3 class="h3">Tools </h3>
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</div>
</div>
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Grid </h3>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</div>
</div>
</div>
```

## JAVASCRIPT

```
var splitterInstance = new ej.Splitter($("#splitter"), {
height: 280, width: 600,
enableAnimation: true,
animationSpeed: 300
properties: [{}, { collapsible: true}]
});
```

The output for **Splitter** when **enableAnimation** is “True”. Expanding or collapsing the outer pane in the **Splitter** produces the animation effect with the animation speed.



## Adjusting Splitter Size

### Height

The height of **Splitter** can be modified by using the **height** property. The default value for **height** property is **null** in **Splitter**. You can set the **height** property by pixel or percentage values.

### Width

The width of **Splitter** can be modified by using the **width** property. The default value for **width** property is **null** in **Splitter**. You can set the **width** property by pixel or percentage values.

### Max Size

Defines the maximum resizable size of the pane when you resize the **Splitter** widget. The default value of **maxSize** is **null** in **Splitter**. You can set the **maxSize** property by pixel values.

### Min Size

Defines the minimum resizable size of the pane when you resize the **Splitter** widget. The default value of **minSize** is **10** in **Splitter**. You can set the **minSize** property by pixel values.

### Pane Size

Defines the pane size in the **Splitter** widget. The default value of **paneSize** is **0px** in **Splitter**. You can set the **paneSize** property by pixel or percentage values.

### Resizable

Defines whether the pane in the **Splitter** is resizable or not. Setting the **resizable** property as **"False"** disables the resize option to the pane. The default value of **resizable** property is **true** in **Splitter**.

The following steps explain the implementation of **Splitter** properties.

In the **HTML** page set the **<div>** element for rendering **Splitter** widget.

### HTML

```
<div id="splitter">
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Tools </h3>
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</div>
</div>
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Grid </h3>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</div>
</div>
</div>
```

### JAVASCRIPT

```
var splitterInstance = new ej.Splitter($("#splitter"), {
height: 280, width: 600,
properties: [{}, { collapsible: true, minSize: 100, maxSize: 800,
paneSize: 300, resizable: true }]
});
```

The output for **Splitter** after adding the properties.

**Tools**

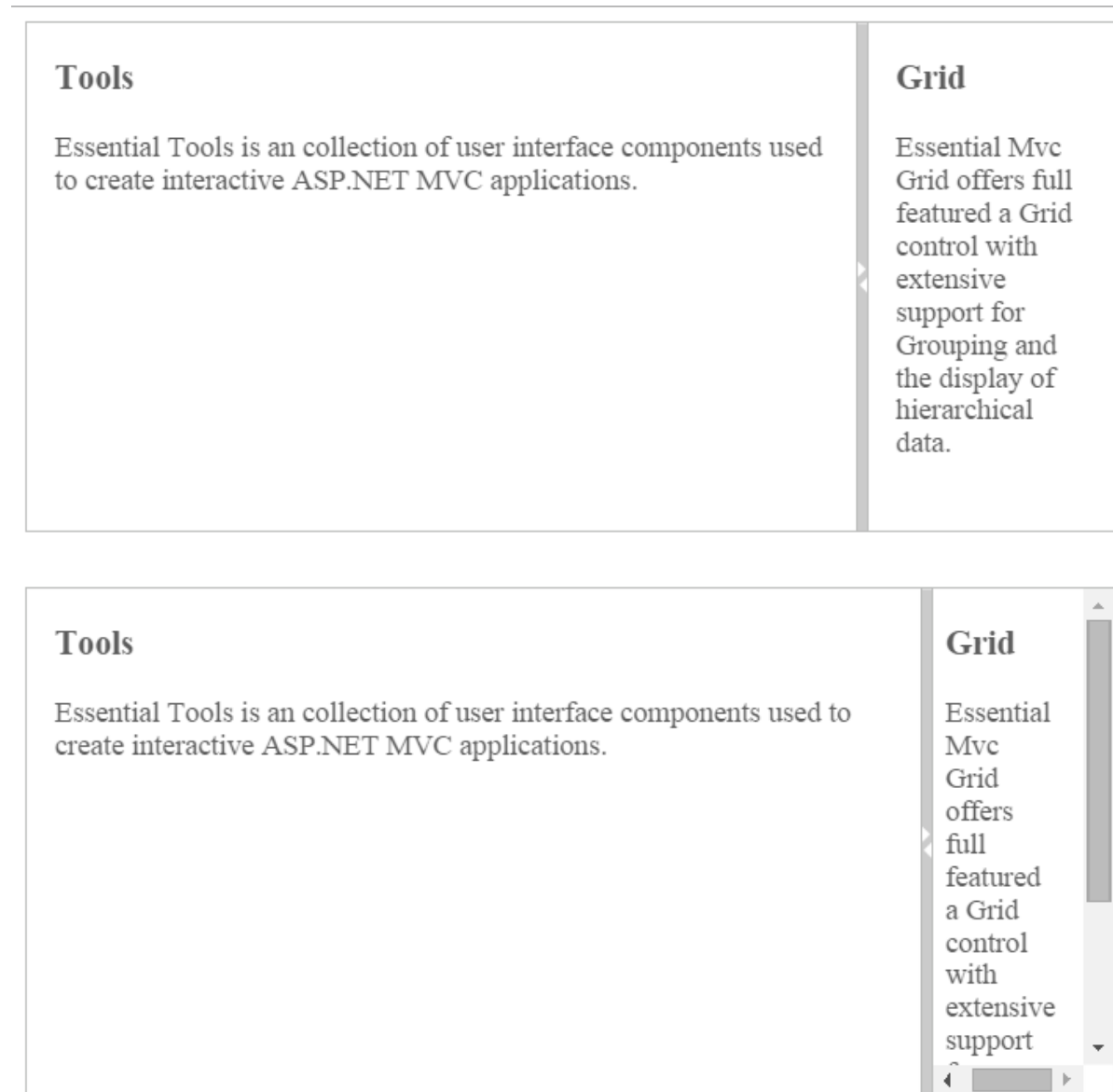
Essential Tools is an collection of user interface components used to create interactive ASP.NET MVC applications.

**Grid**

Essential Mvc Grid offers full featured a Grid control with extensive support for Grouping and the display of hierarchical data.

**Tools**

Essential Tools is an collection of user interface components used to create interactive ASP.NET MVC applications.



### Theme

**Splitter** control's style and appearance can be controlled based on CSS classes. In order to apply styles to the **Splitter** control, refer 2 files namely: **ej.widgets.core.min.css** and **ej.theme.min.css**. If the file **ej.widgets.all.min.css** is referred, then it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.widgets.all.min.css** is the combination of these two.

By default, there are 16 themes support available for Autocomplete control namely

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure

- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark *high-contrast-01 high-contrast-02 material office-365*

### CSS Class

The CSS properties can be customized by using **cssClass** in the **Splitter** widget. The following steps explain the implementation of **cssClass** option in the **Splitter** widget.

In the **HTML** page set the corresponding **<div>** element for rendering **Splitter** control.

### HTML

```
<div id="splitter">
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Tools </h3>
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</div>
</div>
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Grid </h3>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</div>
</div>
</div>
```

### JAVASCRIPT

```
var splitterInstance = new ej.Splitter($("#splitter"), {
height: 280, width: 600,
cssClass: "customCSS"
});
```

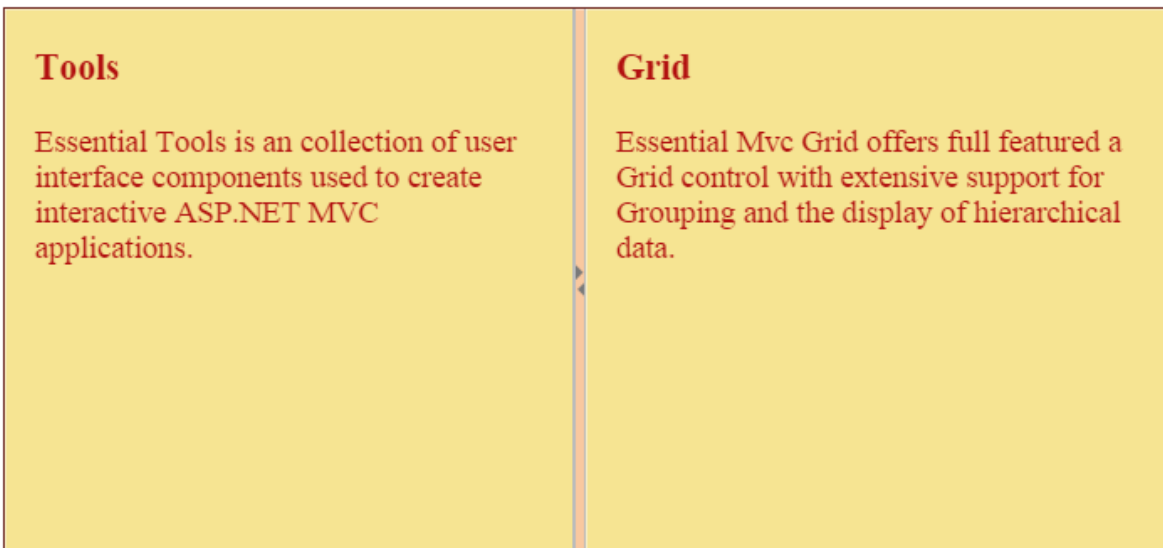
Customize the **CSS** class by setting **CSS** properties.

### CSS

```
<style>
.customCSS {
border-color: #661e19;
}
/*Customize Splitbar*/
.customCSS .e-splitbar {
background-color: #f9c89f;
}
/*Customize Splitter pane*/
.customCSS .e-pane {
color: #b21010;
```

```
background-color: #f6e492;
}
</style>
```

The output for **Splitter** after customizing the CSS class.



### Keyboard Navigation

With the keyboard navigation enabled in the **Splitter** control, it is possible to control the actions of the **Splitter** with the provided shortcut keys. Almost all the **Splitter** actions that are done by mouse can be controlled with shortcut keys.

The various keyboard shortcuts available within the **Splitter** control are discussed in the following table.

#### Keyboard Shortcuts

| Shortcut Key | Description  |
|--------------|--|
| Left         | Moves the Split bar left.                          |
| Right        | Moves the Split bar right.                         |
| Ctrl + Left  | Collapses the left pane.                           |
| Ctrl + Right | Collapses the right pane.                          |
| Up           | Moves the Split bar up.                            |
| Down         | Moves the Split bar down.                          |
| Ctrl + Up    | Collapses the top pane.                            |
| Ctrl + Down  | Collapses the bottom pane.                         |
| Enter        | Resize the pane to the current Split bar position. |
| Esc          | Focuses out from the Split bar.                    |



## Configuring Keyboard Navigation

The following steps explain to enable keyboard interaction for **Splitter** widget.

In the **HTML** page set the corresponding **<div>** element for rendering **Splitter** control.

### HTML

```
<div id="splitter">
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Tools </h3>
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</div>
</div>
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Grid </h3>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</div>
</div>
</div>
```

### JAVASCRIPT

```
module SplitterComponent {
$(function () {
var splitterInstance = new ej.Splitter($("#splitter"), {
height: 280,
width: 600
});
//Control focus key
$(document).on("keydown", function (e) {
if (e.altKey && e.keyCode === 74) { // j- key code.
$("#splitter .e-splitbar")[0].focus();
}
});
});
}
```

Run the sample and press **Alt + J** to focus the **Splitter** widget. Perform provided functionality by using keyboard shortcuts.



## RTL Support

The **Splitter** provides you with **RTL (Right-To-Left)** support. The alignment of **Splitter** can be changed from **Left-To-Right** to **Right-To-Left**.

### Enable RTL

The following steps explain enabling the right-to-left property for **Splitter** widget.

In the **HTML** page set the corresponding **<div>** elements for outer and inner **Splitter** control.

### HTML

```
<div id="outersplitter">
<div>
<div style="padding: 0px 15px;">
<h3 class="h3"> JavaScript </h3>
</div>
</div>
<div id="innersplitter">
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Tools </h3>
Essential Tools is an collection of user interface components used to
create interactive
JavaScript applications.
</div>
</div>
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Chart </h3>
Essential Chart is a business-oriented charting component.
</div>
</div>
<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Grid </h3>
Essential JavaScript Grid offers full featured a Grid control with
extensive support for
```

Grouping and the display of hierarchical data.

```
</div>
</div>
</div>
</div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SplitterComponent {
    $(function () {
        var splitterInstance = new ej.Splitter($("#outterSplitter"), {
            height: 250, width: 600,
            orientation: ej.Orientation.Vertical,
            enableRTL: true,
            properties: [{ paneSize: 60 }]
        });
        var splitterInstance1 = new ej.Splitter($("#innerSplitter"), {
            width: 600,
            properties: [{ paneSize: 200 }, { paneSize: 170 }]
        });
    });
}
```

The output for **Splitter** when **enableRTL** is "true".

| JavaScript  |  |   |
|---|--|---|
| Tools   | Chart  | Grid  |
| Essential Tools is an collection of user interface components used to create interactive .JavaScript applications | Essential Chart is a business-oriented .charting component | Essential JavaScript Grid offers full featured a Grid control with extensive support for Grouping and the display of hierarchical .data |

## Spreadsheet

### Overview

The Spreadsheet control is a Microsoft Excel-like Spreadsheet component for web. It provides editing experience that is very similar to that of excel and it is able to import and export Excel workbook files. The Spreadsheet control includes all the important features of Microsoft Excel like editing, sorting, filtering, formulas, data validation, formatting, table, charts, import and export.

### Key Features

The key features of Spreadsheet control for typescript are:

- **Editing** - Provides support to edit the cell content.
- **Sorting** - Helps you to visualize and figure out your data better, organize and find the data quickly that you want.
- **Filtering** - Offers Excel-like filtering to filter data.
- **Formulas** - Allows you to handle calculations in cells.
- **Table** - Allows you to manage and analyze a group of related data.
- **Formatting** - Provides various formatting support like cell formatting, number formatting, text formatting.
- **Data Validation** - Helps you to restrict the users to enter the valid data in a range.
- **Conditional Formatting** - Allows you to highlight range of cells with a certain colors, depending on the provided conditions.
- **Chart** – Allows you to have graphical representation of data.
- **Fill Series** - Allows you to drag fill data based on adjacent cell data.
- **Printing** – Allows you to print whole sheet / selected range of data.
- **Import and Export** - Allows you to import excel files to view and export Spreadsheet as excel files.

## Getting started

This section explains you the steps required to populate the Spreadsheet with data, format, and export it as excel file. This section covers only the minimal features that you need to know to get started with the Spreadsheet.

### Adding Script Reference

You can create a TypeScript application with the help of the given [Typescript Getting Started Documentation](#).

In your TypeScript app folder, create “app.ts” file.

To get intellisense support and compile time type-checking, we need to follow the below steps,

1. Copy the `ej.web.all.d.ts` file from the below location into your project,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

2. Refer the file in your typescript application(app.ts file).

It is also necessary to refer `jquery.d.ts` in typescript application(app.ts file), which can be downloaded [here](#).

Add the scripts references in the order mentioned in the following code example.

### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
```

```

<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
<script
src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.validate.min.js"></script>
<script src="http://js.syncfusion.com/demos/web/scripts/xljsondata.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{ site.releaseversion
}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>

```

In the above code, `ej.web.all.min.js` script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

**Note:**

- For details about Spreadsheet internal and external dependencies refer following [link](#)

### Initialize Spreadsheet

Add a `div` container to render the Spreadsheet.

#### HTML

```

<!DOCTYPE html>
<html>
<body>
<div id="Spreadsheet"></div>
</body>
</html>

```

Initialize the Spreadsheet in `app.ts` file by using the `ej.Spreadsheet` method. The Spreadsheet is rendered based on default width and height. You can also customize the Spreadsheet dimension by setting the `width` and `height` property in `scrollSettings`.

#### JS

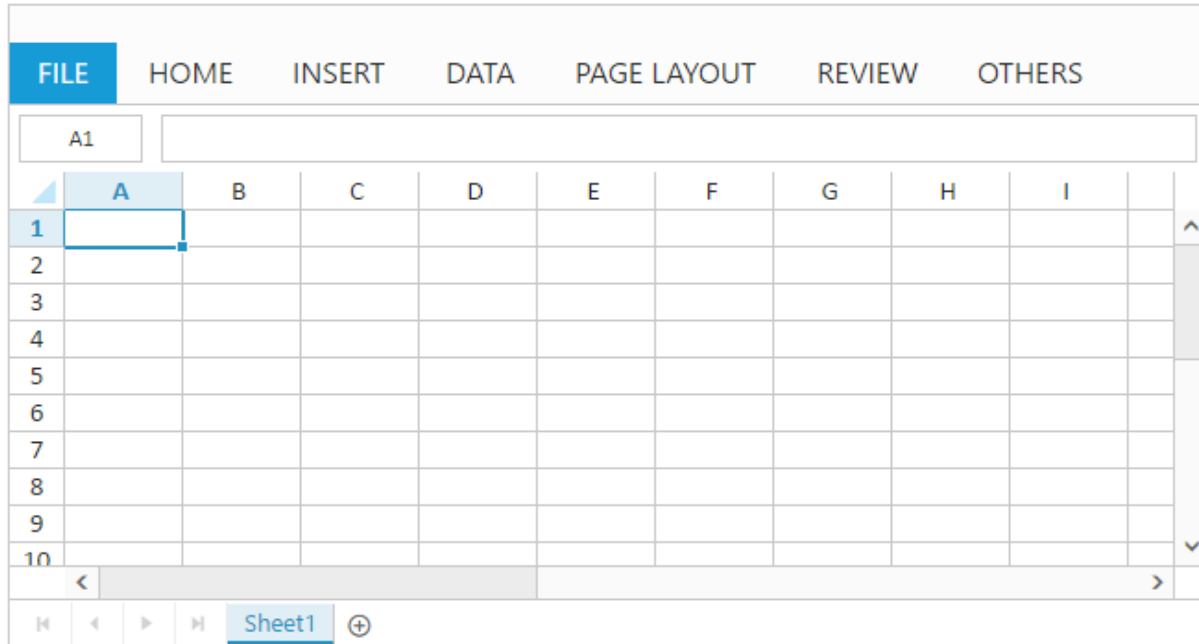
```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module SpreadsheetComponent {
$(function () {
var sample = new ej.Spreadsheet($("#Spreadsheet"), {});
});
}

```

Finally build your application, so that the “app.js” file is automatically generated and got added to your project (User have nothing to do with this file). Now, whatever code changes that you make in app.ts file will be reflected in app.js file automatically.

Now, the Spreadsheet is rendered with default row and column count.



### Populate Spreadsheet with data

Now, this section explains how to populate JSON data to the Spreadsheet. You can set `dataSource` property in `sheet` settings to populate JSON data in Spreadsheet.

### JS

```
module SpreadsheetComponent {
  $(function () {
    var sample = new ej.Spreadsheet($("#Spreadsheet"), {
      // the datasource "window.defaultData" is referred from
      // 'http://js.syncfusion.com/demos/web/scripts/xljsondata.js'
      sheets: [{ rangeSettings: [{ dataSource: (<any>window).defaultData}] }]
    });
  });
}
```

| FILE HOME INSERT DATA PAGE LAYOUT REVIEW OTHERS |                        |            |             |          |       |        |          |
|---|------------------------|------------|-------------|----------|-------|--------|----------|
| A1  | Item Name              |            |             |          |       |        |          |
|   | A                      | B          | C           | D        | E     | F      | G        |
| 1   | Item Name              | Date       | Time        | Quantity | Price | Amount | Discount |
| 2   | Casual Shoes           | 2/14/2014  | 11:34:32 AM | 10       | 20    | 200    | 1        |
| 3   | Sports Shoes           | 6/11/2014  | 5:56:32 AM  | 20       | 30    | 600    | 5        |
| 4   | Formal Shoes           | 7/27/2014  | 3:32:44 AM  | 20       | 15    | 300    | 7        |
| 5   | Sandals & Floaters     | 11/21/2014 | 6:23:54 AM  | 15       | 20    | 300    | 11       |
| 6   | Flip- Flops & Slippers | 6/23/2014  | 12:43:59 AM | 30       | 10    | 300    | 10       |
| 7   | Sneakers               | 7/22/2014  | 10:55:53 AM | 40       | 20    | 800    | 13       |
| 8   | Running Shoes          | 2/4/2014   | 3:44:34 AM  | 20       | 10    | 200    | 3        |
| 9   | Loafers                | 11/30/2014 | 3:12:52 AM  | 31       | 10    | 310    | 6        |
| 10  | Cricket Shoes          | 7/9/2014   | 11:32:14 AM | 41       | 30    | 1210   | 12       |
| Sheet1  |                        |            |             |          |       |        |          |

### Apply Conditional Formatting

Conditional formatting helps you to apply formats to a cell or range with certain color based on the cells values. You can use `allowConditionalFormats` property to enable/disable Conditional formats.

To apply conditional formats for a range use `cFormatRule` property. The following code example illustrates this,

### JS

```
module SpreadsheetComponent {
  $(function () {
    var sample = new ej.Spreadsheet($("#Spreadsheet"), {
      sheets: [{
        rangeSettings: [{ dataSource: (<any>window).defaultData }],
        cFormatRule: [{ action: ej.Spreadsheet.CFormatRule.GreaterThan, inputs:
          ["10"], color: ej.Spreadsheet.CFormatHighlightColor.RedFill, range:
            "D2:D8" }]
      }]
    });
  });
}
```

| FILE HOME INSERT DATA PAGE LAYOUT REVIEW OTHERS |                        |            |             |          |       |        |          |
|---|------------------------|------------|-------------|----------|-------|--------|----------|
| A1  | Item Name              |            |             |          |       |        |          |
|   | A                      | B          | C           | D        | E     | F      | G        |
| 1   | Item Name              | Date       | Time        | Quantity | Price | Amount | Discount |
| 2   | Casual Shoes           | 2/14/2014  | 11:34:32 AM | 10       | 20    | 200    | 1        |
| 3   | Sports Shoes           | 6/11/2014  | 5:56:32 AM  | 20       | 30    | 600    | 5        |
| 4   | Formal Shoes           | 7/27/2014  | 3:32:44 AM  | 20       | 15    | 300    | 7        |
| 5   | Sandals & Floaters     | 11/21/2014 | 6:23:54 AM  | 15       | 20    | 300    | 11       |
| 6   | Flip- Flops & Slippers | 6/23/2014  | 12:43:59 AM | 30       | 10    | 300    | 10       |
| 7   | Sneakers               | 7/22/2014  | 10:55:53 AM | 40       | 20    | 800    | 13       |
| 8   | Running Shoes          | 2/4/2014   | 3:44:34 AM  | 20       | 10    | 200    | 3        |
| 9   | Loafers                | 11/30/2014 | 3:12:52 AM  | 31       | 10    | 310    | 6        |
| 10  | Cricket Shoes          | 7/9/2014   | 11:32:14 AM | 41       | 30    | 1210   | 12       |

### Export Spreadsheet as Excel File

The Spreadsheet can save its data, style, format into an excel file. To enable save option in Spreadsheet set `allowExporting` option in `exportSettings` as `true`. Since Spreadsheet uses server side helper to save documents set `excelUrl` in `exportSettings` option. The following code example illustrates this,

### JS

```

module SpreadsheetComponent {
  $(function () {
    var sample = new ej.Spreadsheet($("#Spreadsheet"), {
      sheets: [{
        rangeSettings: [{ dataSource: (<any>window).defaultData }],
        cFormatRule: [{ action: ej.Spreadsheet.CFormatRule.GreaterThan, inputs:
          ["10"], color: ej.Spreadsheet.CFormatHighlightColor.RedFill, range:
            "D2:D8" }]
      }],
      exportSettings: {
        excelUrl: "http://js.syncfusion.com/demos/ejservices/api/Spreadsheet/Excel
          Export"
      }
    });
  });
}

```

Use shortcut **Ctrl + S** to save Spreadsheet as excel file.

## SunburstChart

### Overview

Sunburst chart is useful for visualizing hierarchical data. The center circle represents the root level in the hierarchy, with outer circles representing the higher levels of the hierarchy.

Some of the key features are,



- Visualize hierarchical data.
- Data label support for better readability.
- Interactively select or highlight segments.
- Interactive legend.
- Colors of the segments can be customized in the SunburstChart.

## Getting Started

This section explains you the steps required to populate the Sunburst Chart with data, data labels, legend and title. This section covers only the minimal features that you need to know to get started with the Sunburst Chart.

### Create Sunburst Chart

You can easily create the Sunburst Chart widget by using the following steps.

#### Add Libraries

1.First create an TypeScript Project and add the following script reference in the app.ts page

For common getting started of TypeScript , you can refer [here](#).

The default type definition file **ej.web.all.d.ts** needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the **jquery.d.ts** file in your TypeScript application, which can be downloaded from [here](#).

2.Add the following script reference in the HTML page

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

In the above code, **ej.web.all.min.js** script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

### Populate Data source:

The datasource for the Sunburst Chart is populated as a JSON object. The “default\_data” contains the JSON data for rendering the Sunburst Chart as shown in the sample.

### JS

```
var default_data = [
  { Category: "Employees", Country: "USA", JobDescription: "Sales",
    JobGroup: "Executive", EmployeesCount: 50 },
  { Category: "Employees", Country: "USA", JobDescription: "Sales",
    JobGroup: "Analyst", EmployeesCount: 40 },
  { Category: "Employees", Country: "USA", JobDescription: "Marketing",
    EmployeesCount: 40 },
  { Category: "Employees", Country: "USA", JobDescription: "Technical",
    JobGroup: "Testers", EmployeesCount: 55 },
  { Category: "Employees", Country: "USA", JobDescription: "Technical",
    JobGroup: "Developers", JobRole: "Windows", EmployeesCount: 175 },
  { Category: "Employees", Country: "USA", JobDescription: "Technical",
    JobGroup: "Developers", JobRole: "Web", EmployeesCount: 70 },
  { Category: "Employees", Country: "USA", JobDescription: "Management",
    EmployeesCount: 40 },
  { Category: "Employees", Country: "USA", JobDescription: "Accounts",
    EmployeesCount: 60 },
  { Category: "Employees", Country: "India", JobDescription: "Technical",
    JobGroup: "Testers", EmployeesCount: 43 },
  { Category: "Employees", Country: "India", JobDescription: "Technical",
    JobGroup: "Developers", JobRole: "Windows", EmployeesCount: 125 },
  { Category: "Employees", Country: "India", JobDescription: "Technical",
    JobGroup: "Developers", JobRole: "Web", EmployeesCount: 60 },
  { Category: "Employees", Country: "India", JobDescription: "HR
    Executives", EmployeesCount: 70 },
  { Category: "Employees", Country: "India", JobDescription: "Accounts",
    EmployeesCount: 45 },
  { Category: "Employees", Country: "Germany", JobDescription: "Sales",
    JobGroup: "Executive", EmployeesCount: 30 },
  { Category: "Employees", Country: "Germany", JobDescription: "Sales",
    JobGroup: "Analyst", EmployeesCount: 40 },
  { Category: "Employees", Country: "Germany", JobDescription: "Marketing",
    EmployeesCount: 50 },
  { Category: "Employees", Country: "Germany", JobDescription: "Technical",
    JobGroup: "Testers", EmployeesCount: 40 },
  { Category: "Employees", Country: "Germany", JobDescription: "Technical",
    JobGroup: "Developers", JobRole: "Windows", EmployeesCount: 65 },
  { Category: "Employees", Country: "Germany", JobDescription: "Technical",
    JobGroup: "Developers", JobRole: "Web", EmployeesCount: 27 },
  { Category: "Employees", Country: "Germany", JobDescription:
    "Management", EmployeesCount: 33 },
  { Category: "Employees", Country: "Germany", JobDescription: "Accounts",
    EmployeesCount: 55 },
  { Category: "Employees", Country: "UK", JobDescription: "Technical",
    JobGroup: "Testers", EmployeesCount: 45 },
  { Category: "Employees", Country: "UK", JobDescription: "Technical",
    JobGroup: "Developers", JobRole: "Windows", EmployeesCount: 96 },
  { Category: "Employees", Country: "UK", JobDescription: "Technical",
    JobGroup: "Developers", JobRole: "Web", EmployeesCount: 55 },
  { Category: "Employees", Country: "UK", JobDescription: "HR Executives",
    EmployeesCount: 60 },
```

```
{ Category: "Employees", Country: "UK", JobDescription: "Accounts",
  EmployeesCount: 30 },
{ Category: "Employees", Country: "France", JobDescription: "Technical",
  JobGroup: "Testers", EmployeesCount: 40 },
{ Category: "Employees", Country: "France", JobDescription: "Technical",
  JobGroup: "Developers", JobRole: "Windows", EmployeesCount: 65 },
{ Category: "Employees", Country: "France", JobDescription: "Technical",
  JobGroup: "Developers", JobRole: "Web", EmployeesCount: 27 },
{ Category: "Employees", Country: "France", JobDescription: "Marketing",
  EmployeesCount: 50 }
];
```

## Initialize Sunburst Chart

1. Add a **div** element that acts as a container for **SunburstChart** widget.

### HTML

```
<body>
<div id="Sunburst"></div>
</body>
```

2. Initialize the Sunburst Chart in ts file by using the ej.SunburstChart method.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SunburstComponent {
  $(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"));
  });
}
```

3. Now, bind the default\_Datasource to **datasource** property of the Sunburst Chart.  
The **levels** property determines the number of hierarchical levels. Each hierarchy level is formed based on the property specified in **groupMemberPath** property, and each arc segment size is calculated using **valueMemberPath**.

### JAVASCRIPT

```
module SunburstComponent {
  $(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
      dataSource: default_data,
      valueMemberPath: "EmployeesCount",
      levels: [
        { groupMemberPath: "Country" },
        { groupMemberPath: "JobDescription" },
        { groupMemberPath: "JobGroup" },
        { groupMemberPath: "JobRole" }
      ],
    },
```

```
});  
});  
}
```

3. The final HTML file appears as follows

### HTML

```
<body>  
<div id="Sunburst"></div>  
</body>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module sunburstComponent {  
  $(function () {  
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {  
      dataSource: default_DataSource,  
      valueMemberPath: "EmployeesCount",  
      levels: [  
        { groupMemberPath: "Country" },  
        { groupMemberPath: "JobDescription" },  
        { groupMemberPath: "JobGroup" },  
        { groupMemberPath: "JobRole" }  
      ],  
    });  
  });  
}
```

### Add Title to the Sunburst Chart

The title of the Sunburst chart is used to provide quick information to the user about the data being plotted in the Sunburst Chart. You can add it by using the `text` property of the `title`

### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {  
    // ...  
    title: {text: "Employees Count"},  
    // ...  
  });  
});
```

### Enable Legend

You can enable or disable the legend by using the `visible` property present inside the `legend`

### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {  
    // ...
```

```

legend:{visible:true ,position:"left"},
// ...
});
});

```

### Add Data Labels

The data labels are used to improve the readability of the Sunburst chart. This can be achieved by enabling the `visible` property in the `dataLabelSettings`.

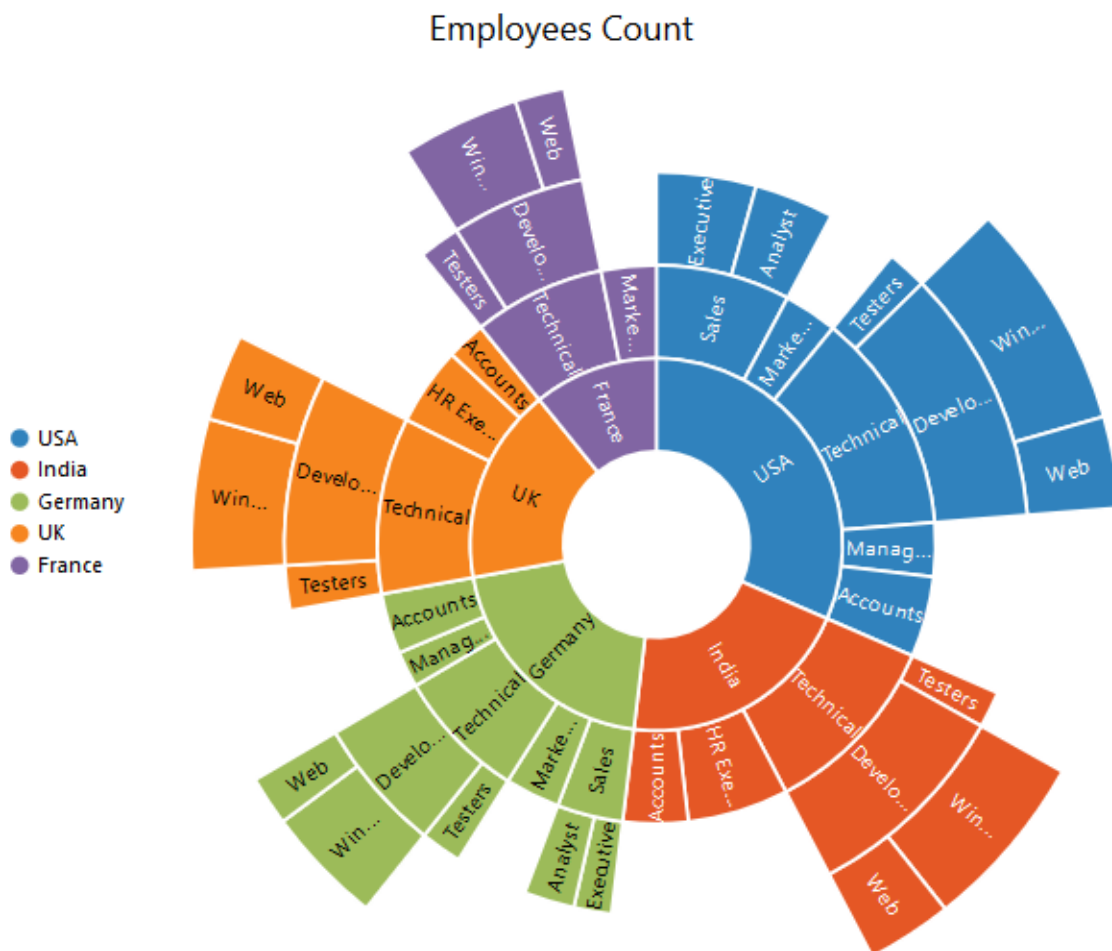
### JAVASCRIPT

```

$(function () {
var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
// ...
dataLabelSettings:{visible:true},
// ...
});
});

```

Now the Sunburst Chart is rendered along with the specified customizations



## Sunburst Elements

The Sunburst region represents the entire chart and all its elements. It includes all the chart elements like Legend, DataLabel, Levels etc. The major properties of the Sunburst Chart are as follows

- **datasource** – Provides the data that are used to generate the chart.
- **valueMemberPath** - Property based on the which the data segments are rendered in the Sunburst chart
- **legend** – displays the legend of the Sunburst Chart
- **levels** - displays the hierarchical levels for the chart
- **dataLabel** – displays the data label for the Sunburst Chart

## Start and End Angle

### Start and End Angle

You can change the start and end angle of Sunburst chart using **startAngle** and **endAngle** property as shown in below code

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module sunburstComponent {
  $(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
      //Set startAngle and endAngle to draw the sunburst chart
      startAngle: -90, endAngle: 90
    });
  });
}
```

## Sunburst Radius

The Radius of the Sunburst chart can be customized by using the **radius** property. The default value of radius is 1 and its value ranges between 0 and 1

#### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    //Set radius to the sunburst chart
    radius: 0.8,
  });
});
```



;

### Sunburst Inner Radius

The Inner Radius of the Sunburst chart can be customized by using the

**innerRadius** property. The default value of innerRadius is 0.4 and its value ranges between 0 and 1

#### **JAVASCRIPT**

```
$(function () {  
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {  
    //Set inner radius to the sunburst chart  
    innerRadius: 0.5,  
    //..  
  });  
});
```



;

### Levels

Sunburst chart is used to display hierarchical data. You can add more than one hierarchical data by using the `levels` property of Sunburst chart. Each level of the hierarchy is represented by circle.

The following code snippet illustrates

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module sunburstComponent {
  $(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
      // ...
      levels: [
        {
          //... to represent the hierarchical data in different levels
        },
      ],
      // ...
    });
  });
}
```

### GroupMemberPath

It is the string property that is used to map the group category value in the dataSource .

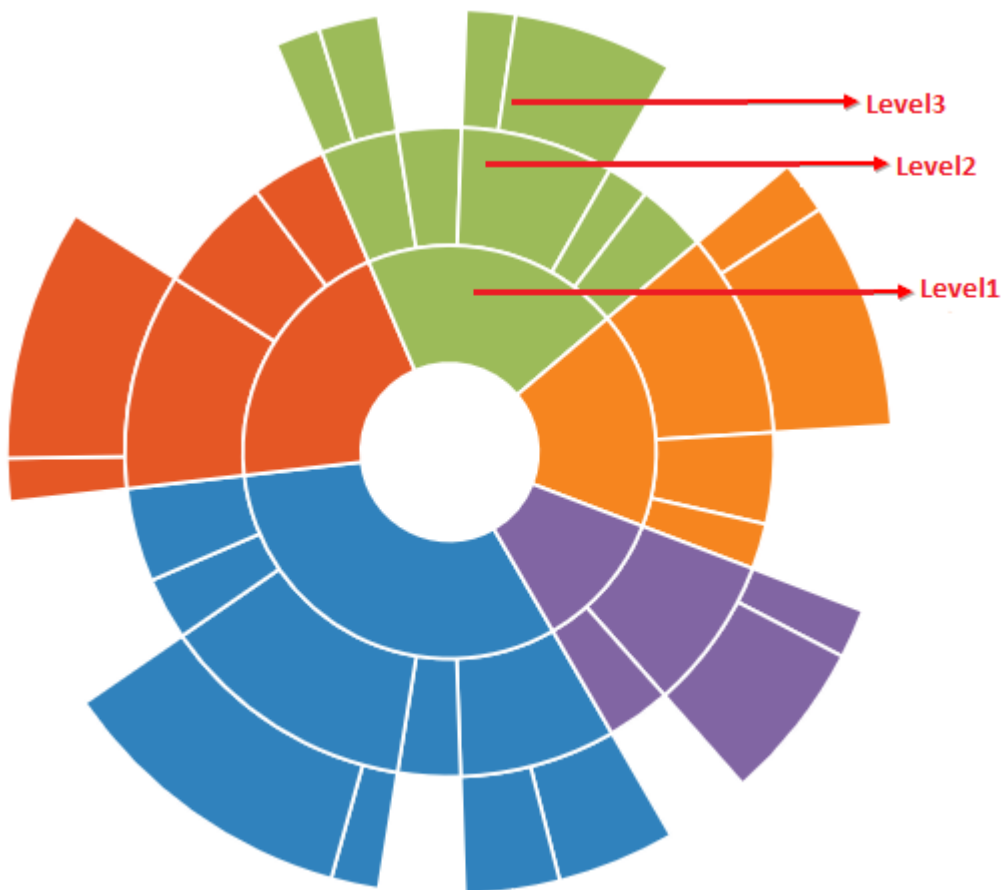
You can define the levels as shown in the below code example



### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {  
    // ...  
    levels: [  
      { groupMemberPath: "Level 1" },  
      { groupMemberPath: "Level 2" },  
      { groupMemberPath: "Level 3" },  
    ],  
    // ...  
  });  
});
```

The following screenshot illustrates the Sunburst Chart with different levels



### Legend

The legend is used to represent the first level of items in the Sunburst Chart. The **legend** can be initialized using the below code snippet

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```

module sunburstComponent {
  $(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
      // ...
      legend: {visible: true},
      // ...
    });
  });
}

```



### Legend Icon

You can specify different shapes of legend icon by using the **shape** property of the legend. By default, legend shape is **Circle**. The Sunburst chart has some predefined shapes such as:

- Circle
- Cross
- Diamond
- Pentagon

- Rectangle
- Triangle

### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    // ...
    legend: {shape: "pentagon"},
    // ...
  });
});
```



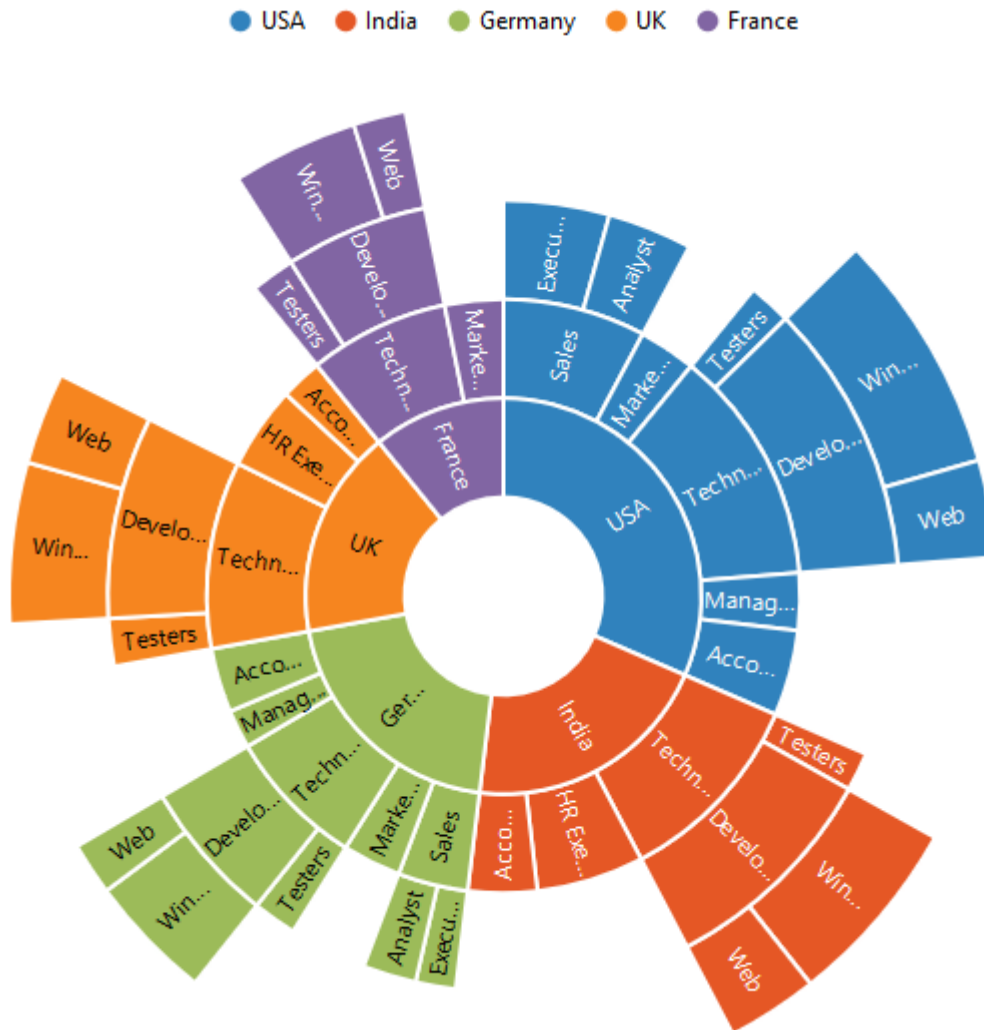
### Positioning the Legend

By using the **position** property, you can position the legend at left, right, top or bottom of the chart.

### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    // ...
    legend: {position: "top"},
    // ...
  });
});
```

```
});
});
```



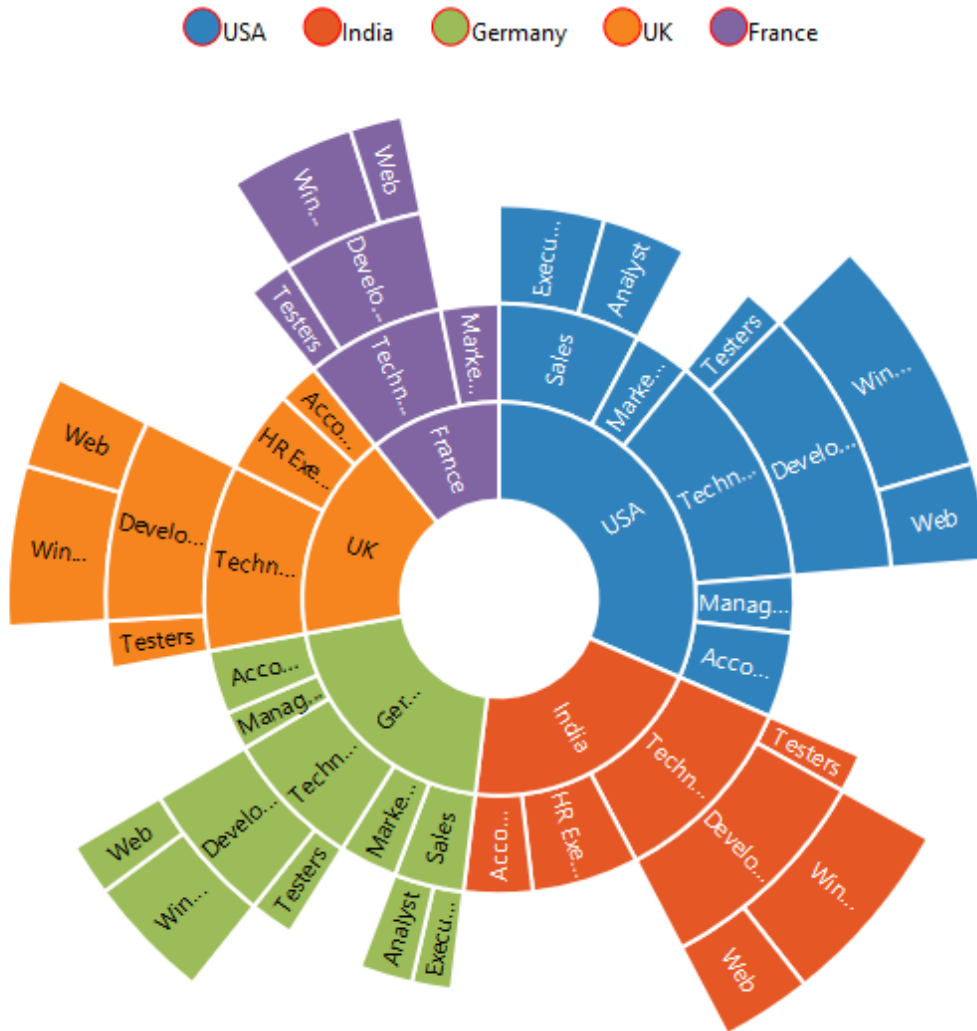
### Customization

#### Legend Item Size and border

You can change the size of the legend items by using the **itemStyle.width** and **itemStyle.height** properties. To change the legend item border, use **border** property of the legend .

#### JAVASCRIPT

```
$(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
        // ...
        legend: { position: "top", itemStyle: { height: 13, width: 13 }, border: { color:
            "#FF0000", width: 1 } },
        // ...
    });
});
```



### Legend Size

By default, legend takes 20% of the height horizontally when it was placed on the top or bottom position and 20% of the width vertically while placing on the left or right position of the chart. You can change this default legend size by using the **size** property of the legend.

### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    // ...
    legend: {position:"top",size:{ height:"75",width:"200"}},
    // ...
  });
});
```



### Legend Row and Columns

You can arrange the legend items horizontally and vertically by using the **rowCount** and **columnCount** properties of the legend.

- When only the **rowCount** is specified, the legend items are arranged according to the **rowCount** and number of columns may vary based on the number of legend items.
- When only the **columnCount** is specified, the legend items are arranged according to the **columnCount** and number of rows may vary based on the number of legend items.
- When both the properties are specified, then the one which has higher value is given preference. For example, when the **rowCount** is 4 and **columnCount** is 3, legend items are arranged in 4 rows.
- When both the properties are specified and have the same value, the preference is given to the **columnCount** when it is positioned at the top/bottom position. The preference is given to the **rowCount** when it is positioned at the left/right position.

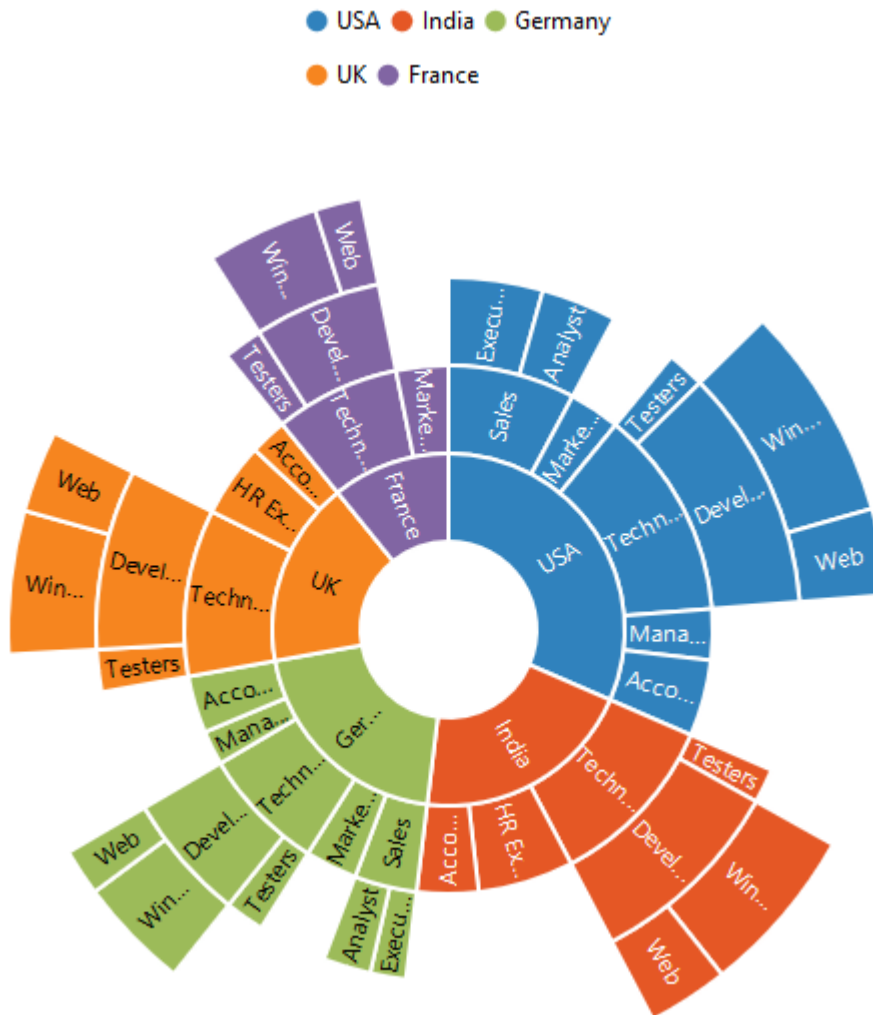
### JAVASCRIPT

```
$ (function () {
```

```

var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
  // ...
  legend: {position:"top",rowCount:"2",columnCount:"3"},
  // ...
});

```



### LegendInteractivity

You can select a specific category while clicking on corresponding legend item through `clickAction` property.

It has three types of action

*ToggleSegmentSelection* *ToggleSegmentVisibility* \* *None*

### ToggleSegmentSelection

Used to highlight specific category while clicking on legend item

### JAVASCRIPT

```

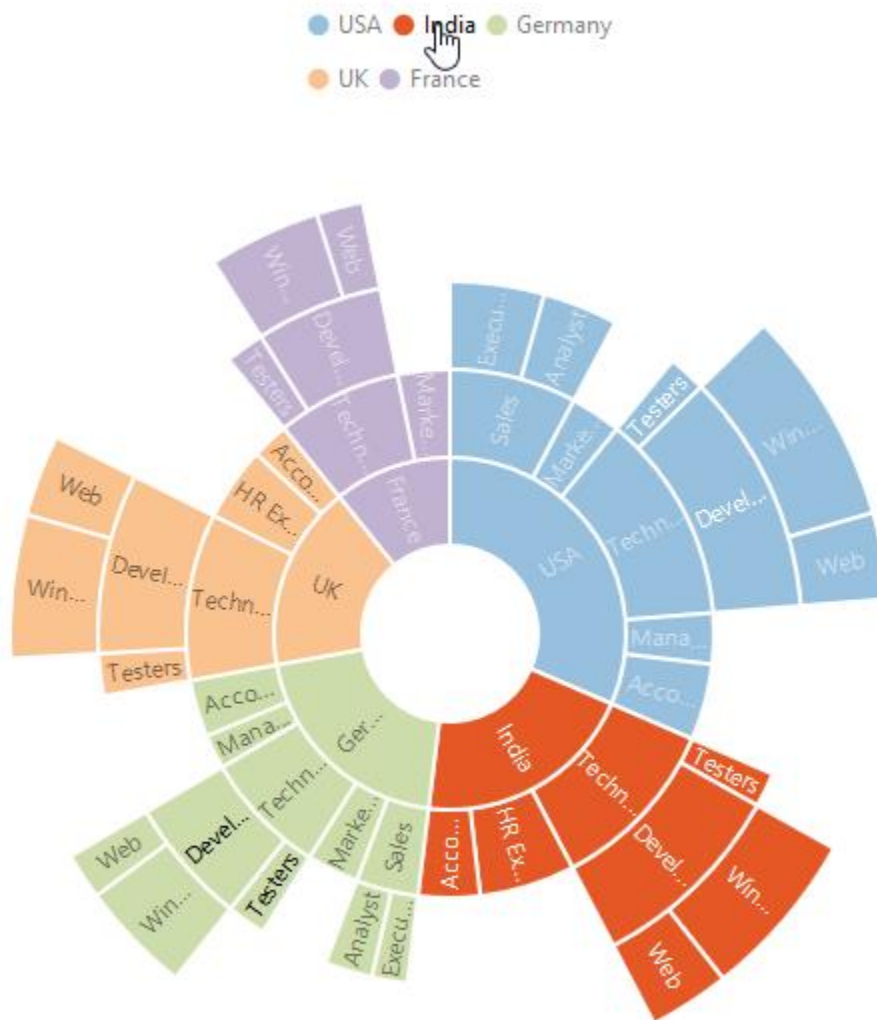
$(function () {

```

```

var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
// ...
legend: {clickAction:"toggleSegmentSelection"},
// ...
});

```



### Toggle Segment Visibility

Used to disable the specific category while clicking on legend item.

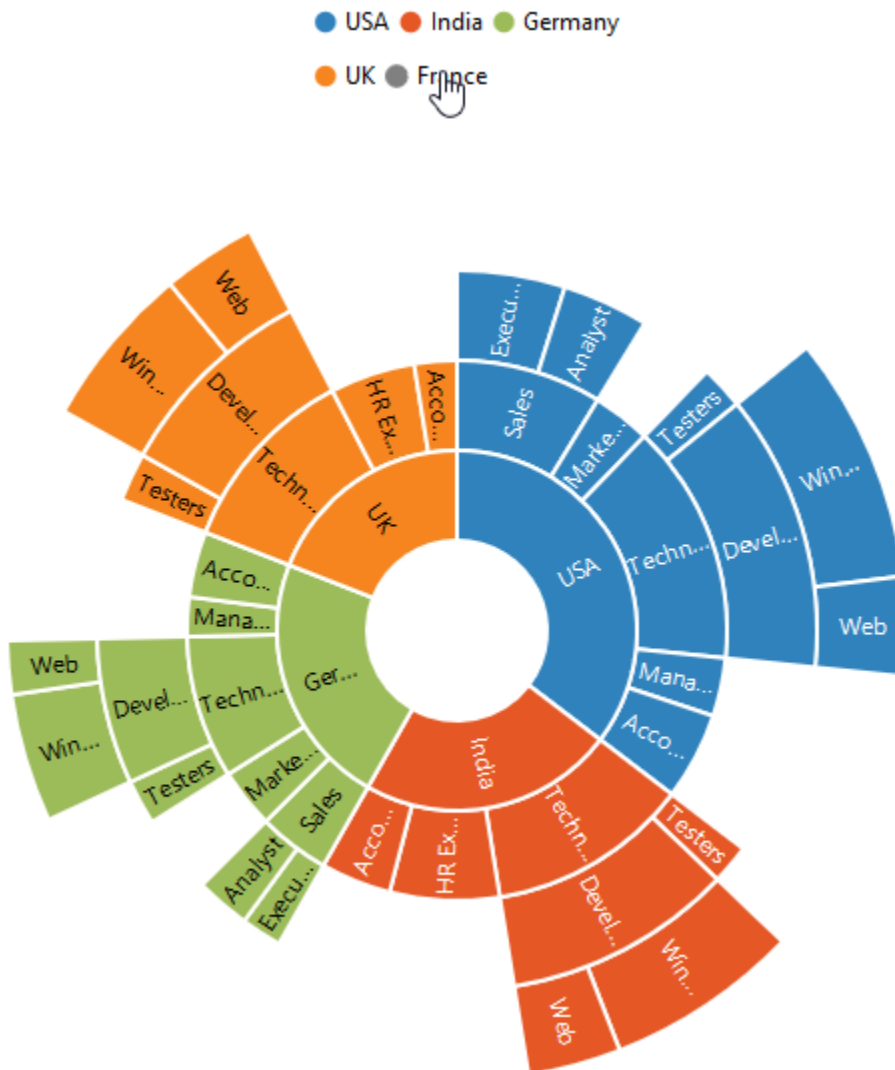
### JAVASCRIPT

```

$(function () {
var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
// ...
legend: {clickAction:"toggleSegmentVisibility"},
// ...
});
});

```





### Data Labels

Sunburst data labels are used to display the data related to the segment. It helps to provide the information about the data points to the users.

You can enable or disable the data labels by setting the **visible** property of the **dataLabelSettings** to true as shown in the below code

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module sunburstComponent {
  $(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
      // ...
      dataLabelSettings: {visible: true},
      // ...
    });
  });
}

```



### Label Overflow mode

When you represent huge data with data labels, they may intersect each other. You can avoid this using the **labelOverflowMode** property.

The following properties are used to avoid the overlapping.

*Trim* – To trim the large data labels. *Hide* – To hide the overlapped data labels.

The following code shows how to set Hide and Trim mode.

### JAVASCRIPT

```
$(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
        // ...
        dataLabelSettings: {visible: true, labelOverflowMode: "hide"},
        // ...
    });
});
```



### JAVASCRIPT

```
$(function () {  
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {  
    // ...  
    dataLabelSettings: {visible: true, labelOverflowMode: "trim"},  
    // ...  
  });  
});
```



### Label Rotation Mode

You can rotate the data label by using **labelRotationMode** property. By default, the labelRotationMode is set as **angle**.

The following code shows how to set labelRotationMode as normal and angle.

#### JAVASCRIPT

```
$(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
        // ...
        dataLabelSettings: {visible: true, labelRotationMode:"normal"},
        // ...
    });
});
```



### JAVASCRIPT

```
$(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
        // ...
        dataLabelSettings: {visible: true, labelRotationMode: "angle"},
        // ...
    });
});
```



### Customizing the data labels

You can customize the appearance of the data point using the **font** property.

#### JAVASCRIPT

```
$(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
        // ...
        dataLabelSettings: {visible: true, font:
        {color:"black",fontWeight:"bold",size:"15px"}},
        // ...
    });
});
```



### Tooltip

**Tooltip** allows you to display any information over a sunburst segment. It appears when mouse hovered over or touch any chart segment. By default, it displays the corresponding segment category name and its value

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module sunburstComponent {
  $(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
      //..
      tooltip: {visible: true},
      //..
    });
  });
}

```



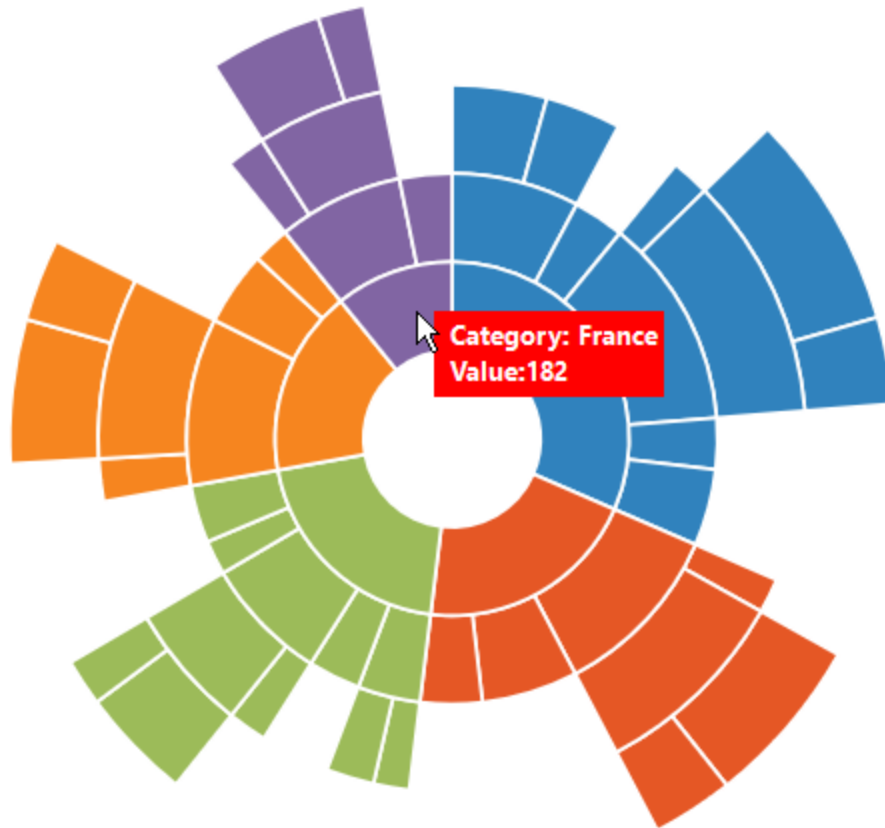
### Tooltip Template

HTML elements can be displayed in the tooltip by using the **template** property of the tooltip. The template property takes the value of the id attribute of the HTML element. You can use the **#point.x#** and **#point.y#** as place holders in the HTML element to display the x and y values of the corresponding point.

### JAVASCRIPT

```
<div id="Tooltip" style="display: none;">
<div id="value" style="background-color:red;padding-top:3px;padding-right:3px">
<div>
<label id="efpercentage" style="color:white">
&#160;&#160;Category:&#160;#point.x#
<br />&#160;&#160;Value:#point.y#
</label>
</div>
</div>
</div>
</div>
$(function () {
var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
//..
tooltip: { visible: true, template:"Tooltip" }
//..
});
});
```





## Highlight

SunburstChart provides highlighting support for the points on mouse hover. To enable the highlighting , set the **enable** property to true in the **highlightSettings**.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module sunburstComponent {
  $(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
      // ...
      highlightSettings: { enable: true },
      // ...
    });
  });
}
```



## Highlight Display mode

You can customize the highlighted segment appearance by using color or opacity. You can choose between color or opacity using the **type** property in the highlight Settings.

*HighlightByColor* – To display the highlighted segment appearance using color. *HighlightByOpacity* – To display the highlighted segment appearance using opacity.

## JAVASCRIPT

```
$(function () {
var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
// ...
highlightSettings: { enable: true, type:"color",color:"red" },
// ...
});
});
```



## Highlight Mode

Sunburst chart provides multiple options to represent the highlighted categories. You can highlight the segment categories by using the **mode** property in highlightSettings

*Child* – To highlight the child of selected parent. *All* – To highlight the entire categories in group. *Parent* – To highlight the parent of selected child. *Single* - To highlight single item in the category.

Child

The following code shows how to set the highlight type as child

## JAVASCRIPT

```
$(function () {
var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
// ...
highlightSettings: { enable: true, mode: "child" },
// ...
});
});
```



### Parent

The parent mode can be enabled by using the below code

### JAVASCRIPT

```
var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
  // ...
  highlightSettings: { enable: true, mode: "parent" },
  // ...
});
```



### Point

To highlight the particular segment, the point mode of the highlight settings is used.

### JAVASCRIPT

```
var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
  // ...
  highlightSettings: { enable: true, mode: "point" },
  // ...
});
```



All

The following code snippet is used for the all mode of highlight settings

#### JAVASCRIPT

```
var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
  // ...
  highlightSettings: { enable: true, mode: "all" },
  // ...
});
```



## Selection

EjSunburstChart provides selection support for the points on mouse click. To enable the selection, set the **enable** property to true in the **selectionSettings**.

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module sunburstComponent {
    $(function () {
        var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
            //...
            selectionSettings: { enable: true },
            //...
        });
    });
}

```



### Selection Display mode

You can customize the selected segment appearance by using color or opacity. You can choose between color or opacity using the **type** property in the selection Settings.

*selectionByColor* – To display the selected segment appearance using color. *selectionByOpacity* – To display the selected segment appearance using opacity.

### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    //...
    selectionSettings: { enable: true, type: "color", color: "red" },
    //...
  });
});
```





### Selection Mode

Sunburst chart provides multiple options to represent the selected categories. You can select the segment categories by using the **mode** property in selectionSettings

*Child* – To selection the child of selected parent. *All* – To selection the entire categories in group. *Parent* – To selection the parent of selected child. *Single* - To selection single item in the category.

### Child

The following code shows how to set the selection type as child

### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    //...
    selectionSettings: { enable: true, mode: "child" },
    //...
  });
});
```



### Parent

The parent mode can be enabled by using the below code

### JAVASCRIPT

```
$(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
        //...
        selectionSettings: { enable: true, mode: "parent" },
        //...
    });
});
```



### Point

To selection the particular segment, the point mode of the selection settings is used.

### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    //..
    selectionSettings: { enable: true, mode: "point" },
    //..
  });
});
```



All

The following code snippet is used for the all mode of selection settings

#### JAVASCRIPT

```
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    //...
    selectionSettings: { enable: true, mode: "all" },
    //...
  });
});
```



## Zooming

Sunburst chart provides zooming (drill down) experience with animation for both mouse and touch enabled devices. It allows you to virtualizing the large sets of data into minimum data view. The zooming is achieved by using the property of **zoomSettings**

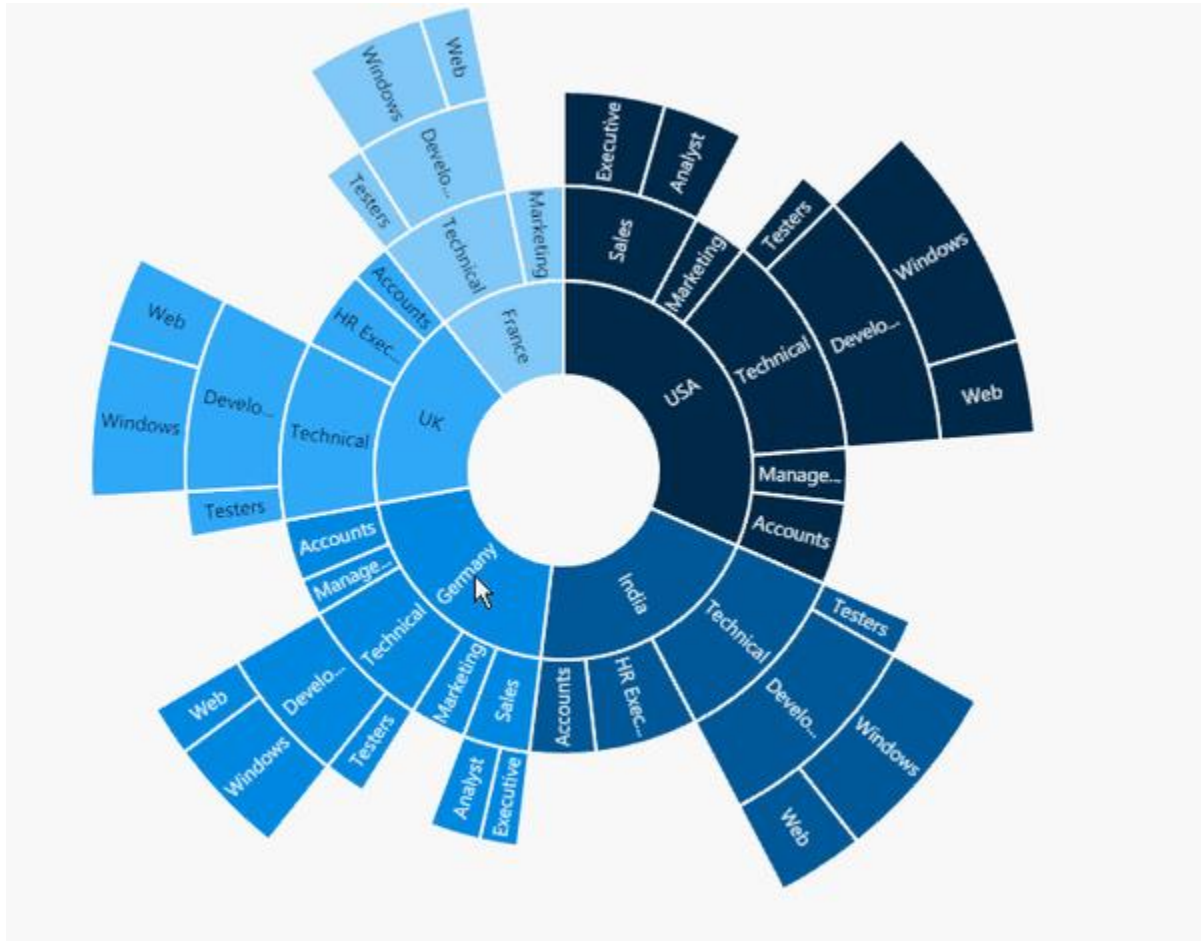
The following code shows how to initialize the zooming.

# JAVASCRIPT

```

/// 
```

```
});
}
```



### Zooming toolbar

By default, zooming toolbar will be enabled while zooming the segment. It contains both back and reset option.

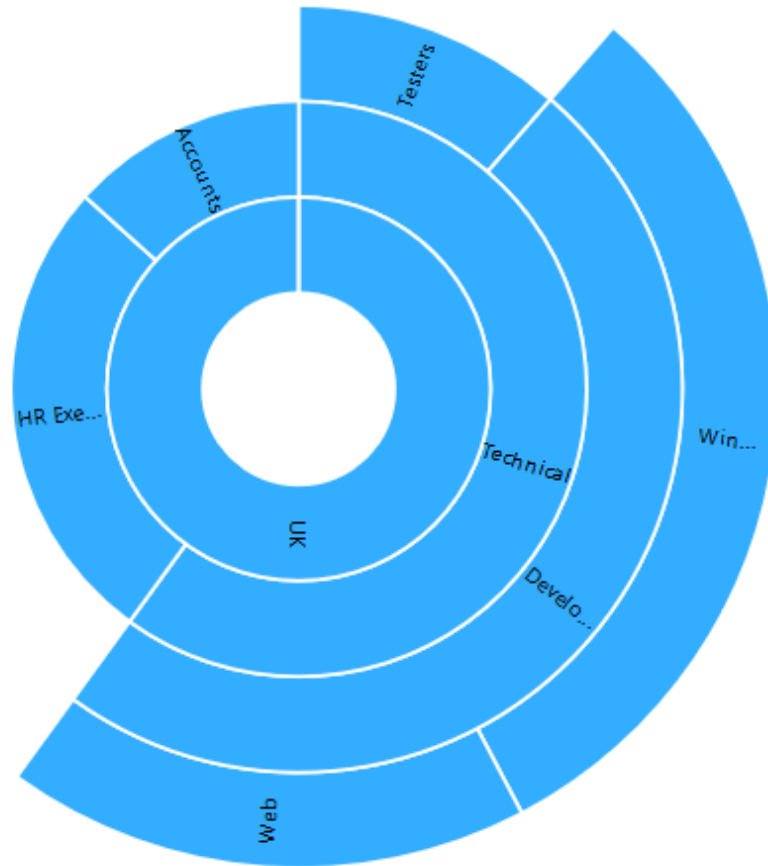
You can align the zooming toolbar position by using **toolbarHorizontalAlignment** and **toolbarVerticalAlignment** property.

### JAVASCRIPT

```
$(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
        //...
        zoomSettings: {enable: true, toolbarHorizontalAlignment: "left"},
        //...
    });
});
```



## Employees Count



### Animation

Sunburst chart allows you to animate the chart segments. You can enable animation using **enableAnimation** property.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module sunburstComponent {
  $(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
      // ...
      enableAnimation: true
      // ...
    });
  });
}
```

### Animation Types

Sunburst chart provide options to animate the chart segments in different ways using **animationType** property.

FadeIn – It gradually changes opacity of the chart segment.

Rotation – During an animation, control rotate from 0 to 360 angle.

#### *Fade In*

The Fade In animation is enabled as follows

#### **JAVASCRIPT**

```
$(function () {  
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {  
    // ...  
    enableAnimation: true,  
    animationType: "fadeIn"  
    // ...  
  });  
});
```

### Population Details

● North America ● South America ● Asia ● Africa ● Europe





### Rotation

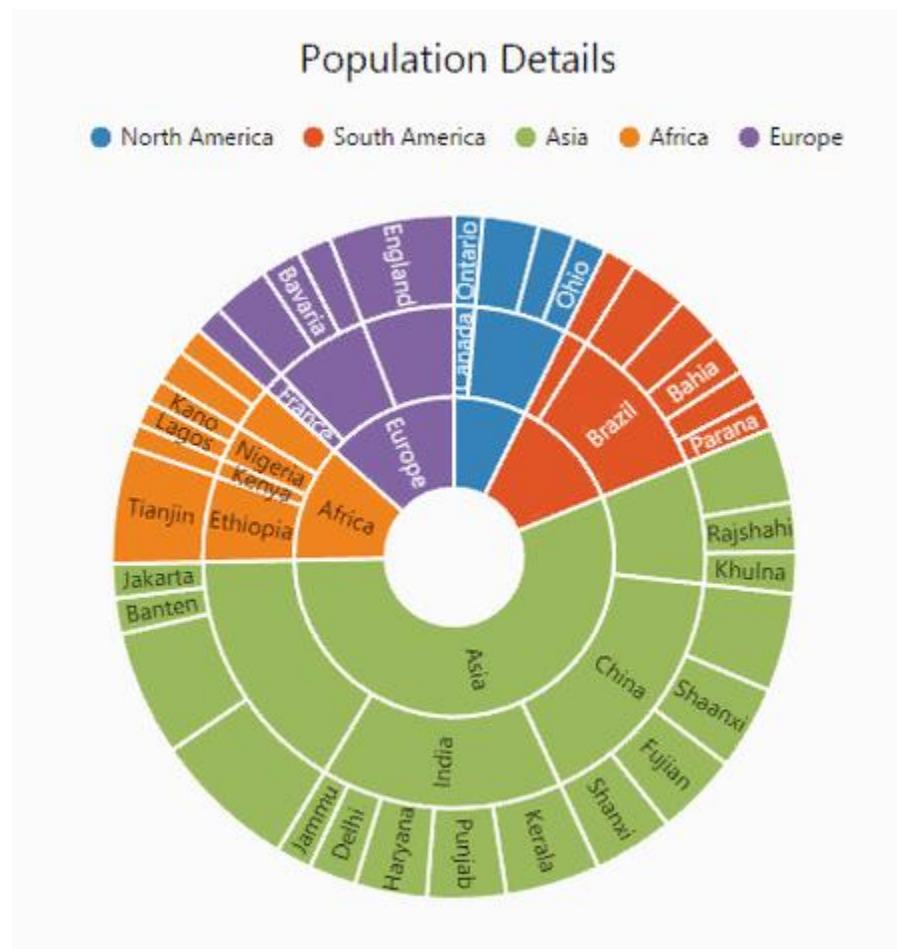
The following example shows how to enable rotation animation in ejSunburstChart

#### JAVASCRIPT

```

$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    // ...
    enableAnimation: true,
    animationType: "rotation"
    // ...
  });
});

```



### Appearance

The appearance of the Sunburst Chart can be customized as shown below

#### Palette

The Sunburst Chart displays different segments in different colors by default. You can customize the color of each segment by providing a custom color palette of your choice by using the **palette** property.

#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />

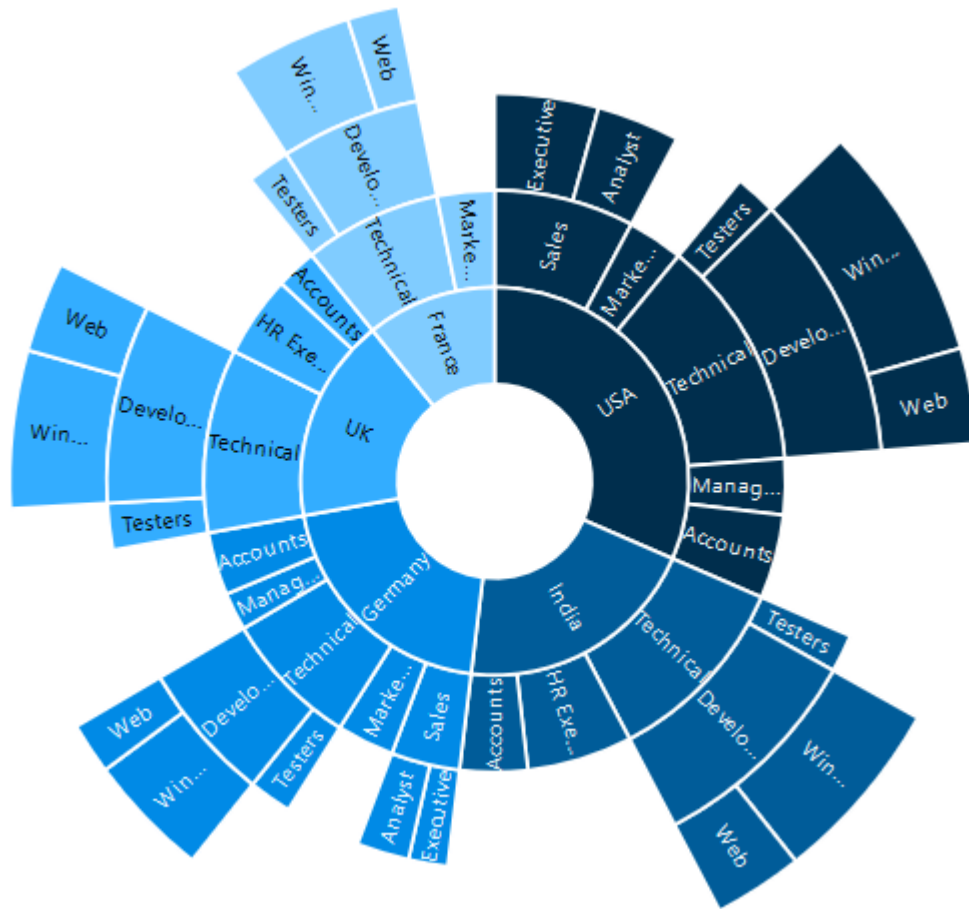
```

```

/// <reference path="tsfiles/ej.web.all.d.ts" />
module sunburstComponent {
$(function () {
var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
// ...
palette: ["#002e4d", "#005c99", "#008ae6", "#33adff", "#80ccff"],
// ...
});
});
}

```

The Sunburst Chart rendered with palette colors



### Built-in Themes

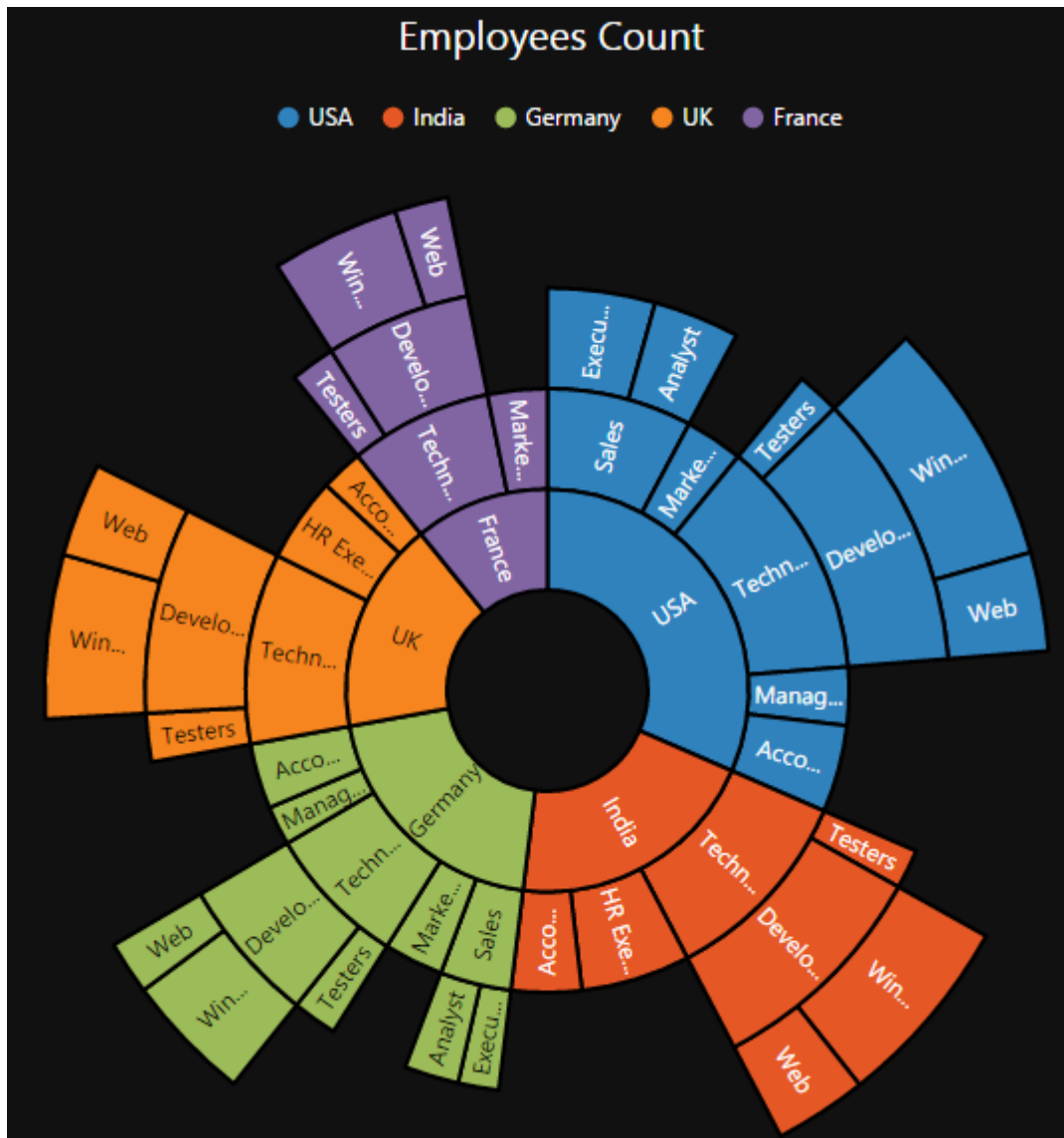
The Sunburst Chart supports different themes.

*flat light* *flat dark* *gradient light* *gradient dark* *azure* *azure dark* *lime* *lime dark* *saffron* *saffron dark* *gradient-azure* *gradient-azure-dark* *gradient-lime* *gradient-lime-dark* *gradient-saffron* *gradient-saffron-dark*

You can set your desired theme by using the **theme** property. **Flat light** is the default theme used in the Sunburst Chart.

### JAVASCRIPT

```
$(function () {
var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
// ...
theme: "saffron",
// ...
});
});
```



## Methods

### [redraw\(\)](#)

**redraw** the entire sunburst. You can call this method whenever you update, add or remove points from the data source or whenever you want to refresh the UI.

### [Example](#)

#### **HTML**

```
<div id="Sunburst">Sunburst</div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SunburstChartComponent {
  $(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
      ///...//
    });
    // Redraws the Sunburst
    sample.redraw();
  });
}
```

*destroy()*

**destroy** the sunburst

Example

## HTML

```
<div id="Sunburst">Sunburst</div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module SunburstChartComponent {
  $(function () {
    var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
      ///...//
    });
    // Destroys the Sunburst
    sample.destroy();
  });
}
```

Events

*load*

Fires before loading, you can use **load** event.

Example

## JAVASCRIPT

```
//load event for sunburst
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    load: function () {
      ///...//
    }
  });
});
```

### *preRender*

Fires before rendering sunburst, you can use **preRender** event.

#### Example

##### JAVASCRIPT

```
//preRender event for sunburst
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    preRender: function () {
      //...//
    }
  });
});
```

### *loaded*

Fires after rendering sunburst, you can use **loaded** event.

#### Example

##### JAVASCRIPT

```
//loaded event for sunburst
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    loaded: function () {
      //...//
    }
  });
});
```

### *dataLabelRendering*

Fires before rendering the data label, you can use **dataLabelRendering**event.

#### Example

##### JAVASCRIPT

```
//data label rendering event for sunburst
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    dataLabelRendering: function () {
      //...//
    }
  });
});
```

### *segmentRendering*

Fires before rendering each segment, you can use **segmentRendering** event.

#### Example

##### JAVASCRIPT

```
//segment rendering event for sunburst
$(function () {
```

```
var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
  segmentRendering: function () {
    //...//
  }
});
```

#### *titleRendering*

Fires before rendering sunburst title, you can use **titleRendering** event.

#### *Example*

##### **JAVASCRIPT**

```
//title rendering event for sunburst
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    titleRendering: function () {
      //...//
    }
  });
});
```

#### *tooltipInitialize*

Fires during initialization of tooltip, you can use **tooltipInitialize** event.

#### *Example*

##### **JAVASCRIPT**

```
//tooltip initialize event for sunburst
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    tooltipInitialize: function () {
      //...//
    }
  });
});
```

#### *pointRegionClick*

Fires after clicking the point in sunburst, you can use **pointRegionClick** event.

#### *Example*

##### **JAVASCRIPT**

```
//point region click event for sunburst
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    pointRegionClick: function () {
      //...//
    }
  });
});
```

#### *pointRegionMouseMove*

Fires while moving the mouse over sunburst points, you can use **pointRegionMouseMove** event.

## Example

**JAVASCRIPT**

```
//point region mouse move event for sunburst
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    pointRegionMouseMove: function () {
      //...//
    }
  });
});
```

*drillDownClick*

Fires when clicking the point to perform drilldown, you can use **drillDownClick** event.

## Example

**JAVASCRIPT**

```
//drill down click event for sunburst
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    drillDownClick: function () {
      //...//
    }
  });
});
```

*drillDownBack*

Fires when resetting drilldown points, you can use **drillDownBack** event.

## Example

**JAVASCRIPT**

```
//drill down back event for sunburst
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    drillDownBack: function () {
      //...//
    }
  });
});
```

*drillDownReset*

Fires after resetting the sunburst points, you can use **drillDownReset** event.

## Example

**JAVASCRIPT**

```
//drill down reset event for sunburst
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    drillDownReset: function () {
      //...//
    }
  });
});
```

```
});
```

### *legendItemRendering*

Fires before rendering the legend item, you can use **legendItemRendering** event.

#### Example

##### **JAVASCRIPT**

```
//legend item rendering event for sunburst
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    legendItemRendering: function () {
      //...//
    }
  });
});
```

### *legendItemClick*

Fires when clicking the legend item, you can use **legendItemClick** event.

#### Example

##### **JAVASCRIPT**

```
//legend item click event for sunburst
$(function () {
  var sample = new ej.datavisualization.SunburstChart($("#Sunburst"), {
    legendItemClick: function () {
      //...//
    }
  });
});
```

## Tab

### Overview

The **Tab** control is an interface where list of items are expanded from a single item. Each **Tab** panel defines its header text or header template, as well as a content template. **Tab** items are dynamically added and removed. **Tabs** can be loaded with AJAX content that is useful for building dashboards where space is limited.

### Key Features

- **Collapsible header:** All headers are collapsible.
- **AJAX load:** Load AJAX content in the Tab content panel.
- **Close button:** Provides a close button for each Tab item.
- **Custom event for expanding header:** Activate a Tab item on a single click, or bind custom events such as mouse over or mouse up to activate a Tab.
- **Header orientation:** Change the header position to the top or bottom of the control.
- **Persist:** Save the current model value to browser cookies to maintain the state.
- **Height style:** Adjust the content panel height.
- **RTL:** Tab headers and content can display right-to-left languages.



- **Add Tab dynamically:** New Tab items can be added and removed at run time.
- **Theme:** Essential JavaScript controls feature 17 built-in themes (6 flat, 6 gradient, bootstrap, 2 high-contrast, material and office-365 effects) and also support the custom skin option to set user-defined themes.
- **Keyboard navigation:** You can interact with the control using the keyboard.

## Getting Started

This section explains briefly about how to create a **Tab** Control in your application with **JavaScript**. The **Essential JavaScript Tab** control is an interface that displays the content in multiple sections. Each **TabPanel** consists of **HeaderText** or **HeaderTemplate** as well as a **ContentTemplate**. **Tab** is useful for a dashboard that is having limited space. The following section guides you to on-demand customize the **Tab** for displaying Hotel menu items, its rating details and ingredients.



### Create Tab Control

The following steps describe the creation of **Tab** control.

Create an **HTML** file and add the following template in the HTML file for creating **Tab** control.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Getting Started - RichTextEditor</title>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"></script>
</head>
</html>
```

Add **Tab** header within <body> tag.

#### HTML

```
<div id="dishType" style="width: 550px">
<ul>
<li><a href="#pizza">Pizza Menu</a></li>
<li><a href="#sandwich">Sandwich Menu</a></li>
<li><a href="#pasta">Pasta Menu</a></li>
</ul>
</div>
```

Initialize the Tab in ts file by using the **ej.Tab** method.

#### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="scripts/typings/ej/ej.web.all.d.ts" />
module TabComponent {
```

```
$(function () {
  var tab = new ej.Tab($("#dishType"));
});
```

The following screen shot illustrates the **Tab** control with Header.



### Configure Content

In this application a detailed description is provided to each item. You can specify the contents in the **Tab** section within the <div> element.

### HTML

```
<div id="pizza" style="background-color: #F5F5F5">
  <!--Food item description-->
  <p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
  poultry, 100% pure mutton, and cheese - and in creating nutritious and
  tasty meals to maintain good health.</p>
</div>
```

You can provide more customization to the **Tab** with **rating** control as content in it for describing the item rating value.

### Create the Rating

The **Essential JavaScript Rating** control provides you an intuitive rating experience that allows you to select the number of stars that represents the rating. For more information about the rating please refer the following link:

<http://help.syncfusion.com/js/rating/getting-started>

The following code example explains you the **rating** control creation. The input element is used to create the **rating** control. Render the input element as **rating** control using the input element id. The code example can be placed within the content description <div> element to declare the rating control and description in the **Tab** section and it can be appended with the header initialization code section <div> element.

### HTML

```
<div id="pizza" style="background-color: #F5F5F5">
  <p>Rating:</p>
  <!--Rating control declaration-->
  <div class="dishRating">
    <input id="pizzaRating" type="text" class="rating" />
  </div>
  <!--Food item description-->
  <p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
  poultry, 100% pure mutton, and cheese - and in creating nutritious and
  tasty meals to maintain good health.</p>
</div>
```

To render the **rating** controls in the first **Tab** element refer the styles mentioned in the following code example.

**CSS**

```
<style type="text/css" class="cssStyles">
.dishRating {
position: absolute;
margin: -31px 0px 0px 80px;
}
</style>
```

The following code example is used for rendering the content with **rating** control within the first **Tab** content section.

**JAVASCRIPT**

```
/// <reference path="jquery.d.ts" />
/// <reference path="scripts/typings/ej/ej.web.all.d.ts" />
module TabComponent {
$(function () {
var tab = new ej.Tab($("#dishType"));
var rating = new ej.Rating($("#pizzaRating"), {
value: 4.9,
precision: ej.Rating.Precision.Exact
});
});
}
```

The following screenshot illustrates the **Tab** content with rating control.



**AJAX Content Load (Load On Demand)**

You can change the contents in sub **Tab** element periodically and you are provided with a support to change the contents without any problems. To achieve the content load, use the Load on Demand concept.

In Load On-Demand, the external HTML file with the necessary details is referred in <href> section during **Tab** header declaration section. When you click the **Tab** header, the AJAX automatically calls the content from the external files and displays in a **Tab** content section.

*Sub Tab with AJAX Content*

Each item is having a variety of options and these options are also specified in the limited space. So you can choose the **Tab** control that is used within the root **Tab** to specify more details.

The following code example illustrates to create the **Tab** control within the root **Tab** element. This HTML code is appended within the previous HTML code section. To render the child **Tab** with its header, add this code example within the first **Tab** <div> element.

**HTML**

```
<!--sub Tab control, the contents loaded with load on demand-->
<div id="pizzaType">
<ul>
<li><span class="cornSpinach"></span>
<a href="http://js.syncfusion.com/UG/Web/Tab-Content/cornSpainach.html">Corn & Spinach </a></li>
<li><span class="chickenDelite"></span>
```

```
<a href="http://js.syncfusion.com/UG/Web/Tab-Content/ChickenDelite.html">Chicken Delite </a></li>
</ul>
</div>
```

The Load On-Demand supported HTML file content (cornSpinach.html)

### HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<div class="e-content">

<div class="ingredients">
Rate : $70<br />
Ingredients : cheese, sweet corn &#38; green capsicums.
<br />
Description: Small pizza bases are topped with spinach and paneer and fresh cream, a nice layer of mozzarella cheese. This is baked until the cheese is all hot and gooey.
</div>
</div>
</body>
</html>
```

The Load On Demand supported HTML file content (chickenDelite.html)

### HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<div class="e-content">

<div class="ingredients">
Rate : $100<br />
Ingredients : cheese, chicken chunks, onions &#38; pineapple chunks.
<br />
Description: This is a tasty, elegant chicken dish that is easy to prepare.
</div>
</div>
</body>
</html>
```

**Note:** In Load On Demand, when the external files are referred from local the following error occurs.

XMLHttpRequest cannot load [http://js.syncfusion.com/UG/Web/Tab-Content/cornSpainach.html?\\_=1399883825133](http://js.syncfusion.com/UG/Web/Tab-Content/cornSpainach.html?_=1399883825133). No 'Access-Control-Allow-Origin' header is present on the requested resource.

To avoid these errors, the external files are hosted in the server to run this application.

The following code example is used to position the image and content in Load On Demand.

### CSS

```
<style type="text/css" class="cssStyles">
/*reuse the previous rating control style section code*/
.ingredients {
height: 180px;
margin-top: 8px;
}
img {
float: left;
margin: 10px 26px 5px 1px;
}
</style>
```

The sub **Tab** control rendering script is represented in the following code example.

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="scripts/typings/ej/ej.web.all.d.ts" />
module TabComponent {
$(function () {
var tab = new ej.Tab($("#pizzaType"));
});
}
```

At the time of AJAX call, the content fetched from external file referenced location is illustrated in the following screenshot.



The following screenshot illustrates the First **Tab** with the sub **Tab** control using Load on Demand.



### Orientation Change

In this application, the sub **Tab** orientation needs to be vertical. By default, **Tab** control renders in horizontal orientation. Change the orientation to vertical using the “**headerPosition**” property. The **Tab** header orientation is set as “**left**”.

The following code example is used to render the sub **Tab** element in the vertical orientation.

### JAVASCRIPT

```
/// <reference path="jquery.d.ts" />
/// <reference path="scripts/typings/ej/ej.web.all.d.ts" />
module TabComponent {
$(function () {
var tab = new ej.Tab($("#dishType"));
var rating = new ej.Rating($("#pizzaRating"), {
value: 4.9,
precision: ej.Rating.Precision.Exact
});
var tabOrientation = new ej.Tab($("#pizzaType"), {
headerPosition: "left",
```

```
height: "221px"
});
});
}
```

The following screenshot illustrates the sub **Tab** with vertical orientation.



### Header Image Customization

In this application, you have to set the **Tab** header image for each Tab item to specify image in <span> tag element during the **Tab** header declaration time.

The following code example is used for customizing the header image.

### CSS

```
<style type="text/css" class="cssStyles">
.dish {
display: inline-block;
vertical-align: middle;
margin: 0px -9px 0px 9px;
}
.pizzaImg {
background: url("http://js.syncfusion.com/UG/Web/Content/rsz_chicken-
delite.png") no-repeat;
height: 25px;
width: 25px;
}
/*reuse the previous header orientation code*/
</style>
```

The following code example is used to add the header image for the root **Tab** header element.

### HTML

```
<div id="dishType" style="width: 550px">
<ul>
<li><span class="dish pizzaImg"></span><a href="#pizza">Pizza
Menu</a></li>
<!-- reuse the remaining tab header -->
</ul>
<!-- reuse the previously defined first tab html content section-->
</div>
```

The following screenshot illustrates the **Tab** with the customized header image.



### Configuring Contents to remaining Tab items

The second and third **Tab** contents are declared in the same method as of the first **Tab** content declaration. These **Tabs** also consist of rating and sub **Tab** controls. The sub **Tab** control contents are loaded in Load On Demand support.

The following code example can be placed within the previous image customization **HTML** code section.

**HTML**

```

<!--reuse the first tab header defined in previous image customization -->
<li><span class="dish sandwichImg"></span><a href="#sandwich">Sandwich
Menu</a></li>
<li><span class="dish pastaImg"></span><a href="#pasta">Pasta
Menu</a></li>

```

Add the second **Tab** contents in <div> element during initialization.

**HTML**

```

<div id="sandwich" style="background-color: #F5F5F5">
<p>Rating:</p>
<!--Rating control declaration-->
<div class="dishRating">
<input id="sandwichRating" type="text" class="rating" />
</div>
<!--dish description-->
<p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
<!--sub Tab control, the contents loaded with load on demand-->
<div id="sandwichType">
<ul>
<li>
<a href="http://js.syncfusion.com/UG/Web/Tab-
Content/gardenVeggie.html">Garden Veggie </a></li>
<li>
<a href="http://js.syncfusion.com/UG/Web/Tab-
Content/chickenTikka.html">Chicken Tikka </a></li>
<li>
<a href="http://js.syncfusion.com/UG/Web/Tab-
Content/paneerTikka.html">Paneer Tikka </a></li>
</ul>
</div>
</div>

```

Add third **Tab** contents in <div> element during initialization.

**HTML**

```

<div id="pasta" style="background-color: #F5F5F5">
<p>Rating:</p>
<!--Rating control declaration-->
<div class="dishRating">
<input id="pastaRating" type="text" class="rating" />
</div>
<!--dish description-->
<p>Pasta cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
<!--sub Tab control, the contents loaded with load on demand-->
<div id="pastaType">
<ul>

```

```

<li>
<a href="http://js.syncfusion.com/UG/Web/Tab-Content/khemmaPasta.html">Kheema Pasta </a></li>
<li>
<a href="http://js.syncfusion.com/UG/Web/Tab-Content/tunaPasta.html">Tuna Pasta</a></li>
<li>
<a href="http://js.syncfusion.com/UG/Web/Tab-Content/channaPasta.html">Channa Pasta
</a></li>
</ul>
</div>
</div>

```

Apply the following styles to the **Tab**.

### CSS

```

<style type="text/css" class="cssStyles">
/*to reuse the previous style section code and following css*/
.sandwichImg, .pastaImg {
height: 25px;
width: 25px;
}
.sandwichImg {
background: url("http://js.syncfusion.com/UG/Web/Content/rsz_garden-fresh.png") no-repeat;
}
.pastaImg {
background: url("http://js.syncfusion.com/UG/Web/Content/rsz_garden-veggie.png") no-repeat;
}
</style>

```

After the content declaration of all the **Tab** control, render the final output using the following code example.

### JAVASCRIPT

```

/// <reference path="jquery.d.ts" />
/// <reference path="scripts/typings/ej/ej.web.all.d.ts" />
module TabComponent {
$(function () {
var tab = new ej.Tab($("#dishType"));
var rating = new ej.Rating($("#pizzaRating"), {
value: 4.9,
precision: ej.Rating.Precision.Exact
});
var tabOrientation = new ej.Tab($("#pizzaType"), {
headerPosition: "left",
height: "221px"
});
var exactRating = new ej.Rating($("#sandwichRating"), {
value: 3.9,
precision: ej.Rating.Precision.Exact
});
});

```



```

var orientationTab = new ej.Tab($("#pastaType"), {
  headerPosition: "left",
  height: "221px"
});
var precisionRating = new ej.Rating($("#pastaRating"), {
  value: 4.5,
  precision: ej.Rating.Precision.Exact
});
});
}

```

The following screenshot illustrates you the second **Tab** contents in **Tab** and the final hotel menu with rating, description and ingredients of the item in the Tab interface.



## Behavior Settings

### Close Button

By default, **Tab** contents are rendered without **Close Button**. You can add the **Close Button** by setting the '**showCloseButton**' property to '**true**'. When you move cursor over the **Tab** headers, the **Close Button** is displayed.

The following code example is used to render the **Tab** widget with **Close Button**.

Add the following **HTML** for simple **Tab** creation with **Close Button**.

### HTML

```

<div id="dish" style="width: 650px">
<ul>
<li><a href="#pizza">Pizza Menu</a></li>
<li><a href="#sandwich">Sandwich Menu</a></li>
</ul>
<div id="pizza" style="background-color: #F5F5F5">
<!--Food item description-->
<p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>
<div id="sandwich" style="background-color: #F5F5F5">
<!--dish description-->
<p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
</div>
</div>

```

### JAVASCRIPT

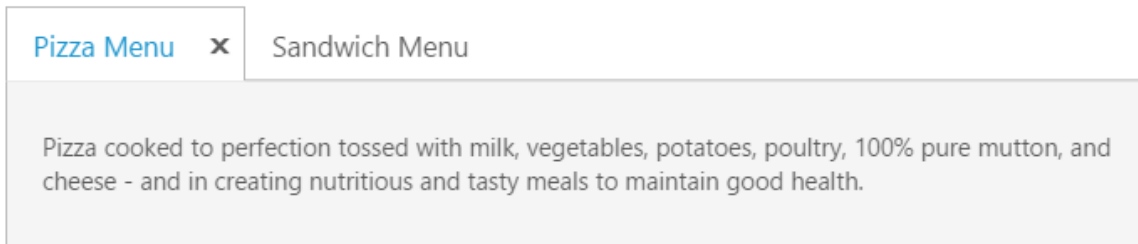
```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
  $(function () {
    var sample = new ej.Tab($("#dish"), {
      showCloseButton: true
    });
  });
}

```

```
});
}
```

The following screenshot illustrates the **Tab** with **Close Button**.



### Orientation

By default, **Tab** control renders in horizontal orientation. You can change the **Orientation** to vertical using the '**headerPosition**' property. Using this property, you can customize the header by "**top**", "**bottom**", "**left**", and "**right**".

The following code example is used to render the sub **Tab** widget in the vertical orientation.

Add the following **HTML** for **Tab** orientation.

### HTML

```
<div id="dish" style="width: 650px">
<ul>
<li><a href="#pizza">Pizza Menu</a></li>
<li><a href="#sandwich">Sandwich Menu</a></li>
</ul>
<div id="pizza" style="background-color: #F5F5F5">
<!--Food item description-->
<p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>
<div id="sandwich" style="background-color: #F5F5F5">
<!--dish description-->
<p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
</div>
</div>
```

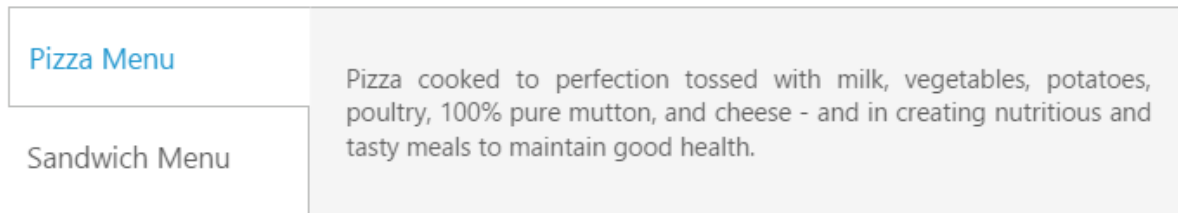
### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
$(function () {
var sample = new ej.Tab($("#dish"), {
headerPosition: "left",
height: "220px"
});
});
});
```

```
}

```

The following screenshot illustrates the sub **Tab** with vertical orientation.



### State Maintenance

When the page gets refreshed or reloaded, the **Tab** state is changed (i.e.) the focus is moved to start **Tab**. You can maintain the state of the **Tab** by using 'enablePersistence' property. When this property is set to 'true', it retains the state.

The following code example is used to render the **Tab** widget with state maintenance.

Add the following **HTML** for **Tab state maintenance**.

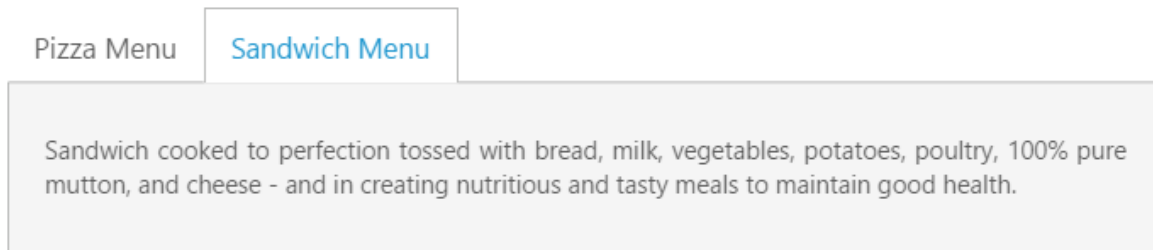
### HTML

```
<div id="dish" style="width: 650px">
<ul>
<li><a href="#pizza">Pizza Menu</a></li>
<li><a href="#sandwich">Sandwich Menu</a></li>
</ul>
<div id="pizza" style="background-color: #F5F5F5">
<!--Food item description-->
<p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>
<div id="sandwich" style="background-color: #F5F5F5">
<!--dish description-->
<p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
</div>
</div>
```

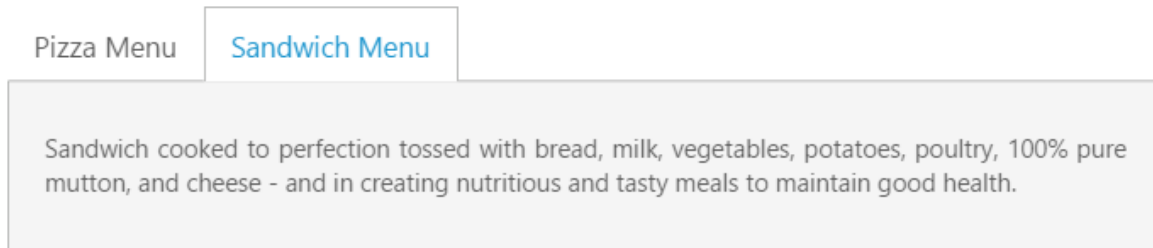
### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
$(function () {
var sample = new ej.Tab($("#dish"), {
enablePersistence: true
});
});
}
```

The following screenshot illustrates the **Tab** with **State maintenance**.



State before page refresh



State after page refresh

## Appearance and Styling

### Header Image Customization

To set the **Tab** header image for each **Tab** item you can specify image in `<span>` tag element during the **Tab** header declaration time.

The following code example is used to add the header image for the root **Tab** header element.

Add the following **HTML** to render **Tab** with header image.

### HTML

```
<div id="dish" style="width: 650px">
<ul>
<li><span class="dish pizzaImg"></span><a href="#pizza">Pizza
Menu</a></li>
<li><span class="dish sandwichImg"></span><a href="#sandwich">Sandwich
Menu</a></li>
</ul>
<div id="pizza" style="background-color: #F5F5F5">
<!--Food item description-->
<p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>
<div id="sandwich" style="background-color: #F5F5F5">
<!--dish description-->
<p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
</div>
</div>
```

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
$(function () {
var sample = new ej.Tab($("#dish"), {
});
});
}

```

Add following **CSS** for header image customization.

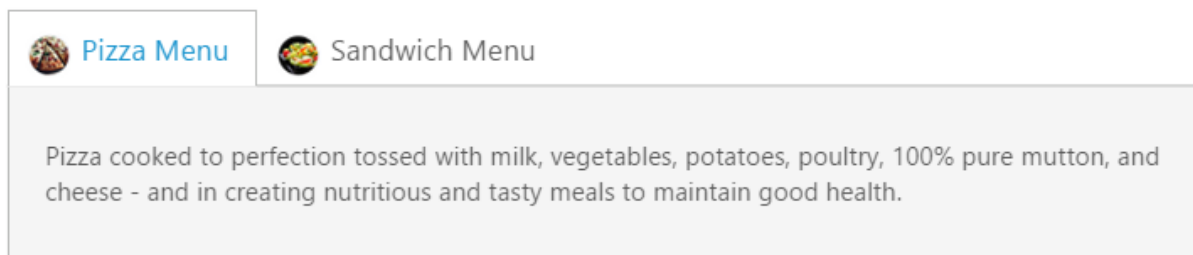
### CSS

```

<style type="text/css" class="cssStyles">
.dish {
display: inline-block;
vertical-align: middle;
margin: 0px -9px 0px 9px;
}
.pizzaImg {
background: url("http://js.syncfusion.com/UG/Web/Content/rsz_chicken-
delite.png") no-repeat;
height: 25px;
width: 25px;
}
.sandwichImg, .pastaImg {
height: 25px;
width: 25px;
}
.sandwichImg {
background: url("http://js.syncfusion.com/UG/Web/Content/rsz_garden-
fresh.png") no-repeat;
}
</style>

```

The following screenshot illustrates the **Tab** with the customized header image.



### Header Image Customization

#### Rounded corner

By enabling '**showRoundedCorner**' property, you can customize the shape of the **Tab** widget from regular rectangular shape to rounded rectangle shape that is set to '**false**' by default.

The following code example is used to render the **Tab** widget with **Rounded Corner**.

Add the following **HTML** to render **Tab** with rounder corner.

**HTML**

```

<div id="dish" style="width: 650px">
<ul>
<li><a href="#pizza">Pizza Menu</a></li>
<li><a href="#sandwich">Sandwich Menu</a></li>
</ul>
<div id="pizza" style="background-color: #F5F5F5">
<!--Food item description-->
<p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>
<div id="sandwich" style="background-color: #F5F5F5">
<!--dish description-->
<p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
</div>
</div>

```

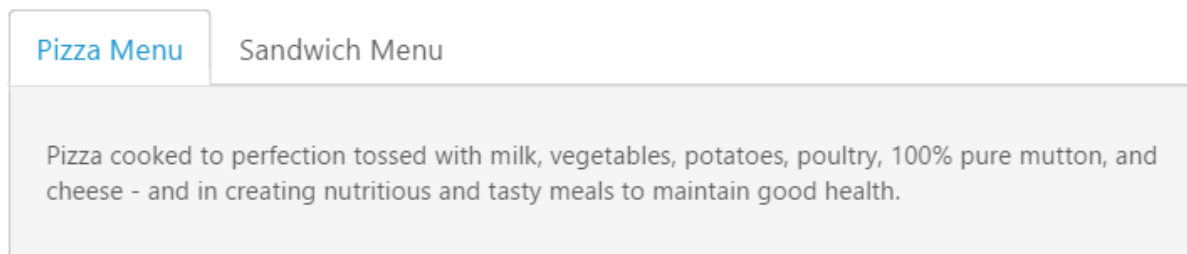
**JAVASCRIPT**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
$(function () {
var sample = new ej.Tab($("#dish"), {
showRoundedCorner: true
});
});
}

```

The following screenshot illustrates the **Tab** with Rounded corner.

**Enable/Disable**

You can enable or disable the **Tab** widget by 'enabled' property. By default, the property set to 'true'.

The following code example is used to render the **Tab** widget with enable/disable.

Add the following **HTML** to render **Tab** with enable/disable.

**HTML**

```

<div id="dish" style="width: 650px">
<ul>
<li><a href="#pizza">Pizza Menu</a></li>

```

```

<li><a href="#sandwich">Sandwich Menu</a></li>
</ul>
<div id="pizza" style="background-color: #F5F5F5">
  <!--Food item description-->
  <p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
  poultry, 100% pure mutton, and cheese - and in creating nutritious and
  tasty meals to maintain good health.</p>
</div>
<div id="sandwich" style="background-color: #F5F5F5">
  <!--dish description-->
  <p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
  potatoes, poultry, 100% pure mutton, and cheese - and in creating
  nutritious and tasty meals to maintain good health.</p>
</div>
</div>

```

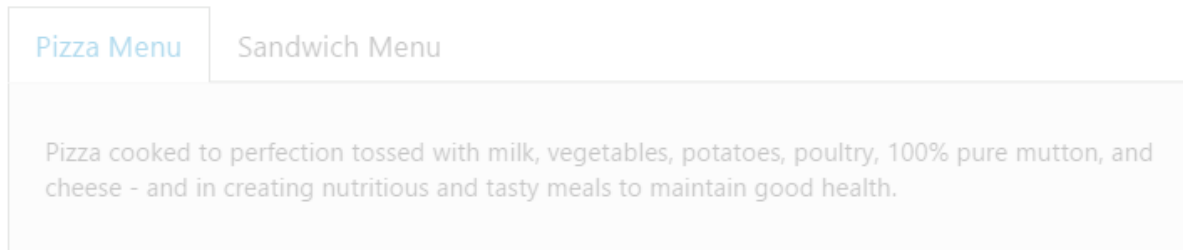
## JAVASCRIPT

```

// Add the following script to render Tab with disabled format.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
  $(function () {
    var sample = new ej.Tab($("#dish"), {
      enabled: false
    });
  });
}

```

The following screenshot illustrates the **Tab** with disabled format.



## Enabling Reload Icon

Without refresh/reload the whole page, you can reload a particular **Tab** using **Reload** icon. The **Reload** icon is appeared at right corner of the **Tab** by enabling the property '**showReloadIcon**' to '**true**'. When you move cursor over the **Tab** headers, the **Reload** icon is displayed. By default the property value is set to '**false**'.

The following code example is used to render the **Tab** widget with **Reload** icon.

Add the following **HTML** to render **Tab** with **Reload** icon.

## HTML

```

<div id="dish" style="width: 650px">
  <ul>
    <li><a href="#pizza">Pizza Menu</a></li>
    <li><a href="#sandwich">Sandwich Menu</a></li>
  </ul>
</div>

```

```

</ul>
<div id="pizza" style="background-color: #F5F5F5">
  <!--Food item description-->
  <p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
  poultry, 100% pure mutton, and cheese - and in creating nutritious and
  tasty meals to maintain good health.</p>
</div>
<div id="sandwich" style="background-color: #F5F5F5">
  <!--dish description-->
  <p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
  potatoes, poultry, 100% pure mutton, and cheese - and in creating
  nutritious and tasty meals to maintain good health.</p>
</div>
</div>

```

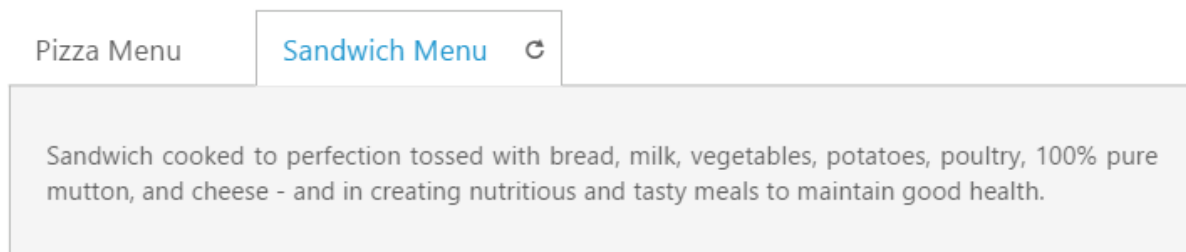
## JAVASCRIPT

```

// Add the following script to render Tab with Reload icon.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
  $(function () {
    var sample = new ej.Tab($("#dish"), {
      showReloadIcon: true
    });
  });
}

```

The following screenshot illustrates the **Tab** with **Reload** icon.



## Collapsible Tabs

You can collapse the **Tab** content by enabling the 'collapsible' property to 'true'. When the property is set to 'true' then click the active **Tab** header, the **Tab** contents are hidden. By default, the property value is set to 'false'.

The following code example is used to render the **Tab** widget with customized collapsible mode.

Add the following **HTML** to render **Tab** with customized collapsible mode.

## HTML

```

<div id="dish" style="width: 650px">
  <ul>
    <li><a href="#pizza">Pizza Menu</a></li>
    <li><a href="#sandwich">Sandwich Menu</a></li>
  </ul>
  <div id="pizza" style="background-color: #F5F5F5">

```



```

<!--Food item description-->
<p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>
<div id="sandwich" style="background-color: #F5F5F5">
<!--dish description-->
<p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
</div>
</div>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
  $(function () {
    var sample = new ej.Tab($("#dish"), {
      collapsible: true
    });
  });
}

```

The following screenshot illustrates the **Tab** with customized collapsible mode.



## Adjusting Tab Size

### Height Adjust Mode and Height

The height of the **Tab** widget is customized by 'height' property. The **Tab** widget height depends on 'heightAdjustMode' property. Using the heightAdjustMode property, you can adjust height by "content", "auto", "fill". By default the heightAdjustMode is set as **content**.

The following code example is used to render the **Tab** widget with customized height and height adjust mode.

Add the following **HTML** to render **Tab** with customized height and height adjust mode.

## HTML

```

<div id="dish" style="width: 650px">
<ul>
<li><a href="#pizza">Pizza Menu</a></li>
<li><a href="#sandwich">Sandwich Menu</a></li>
</ul>
<div id="pizza" style="background-color: #F5F5F5">
<!--Food item description-->
<p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>

```

```

</div>
<div id="sandwich" style="background-color: #F5F5F5">
  <!--dish description-->
  <p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
  potatoes, poultry, 100% pure mutton, and cheese - and in creating
  nutritious and tasty meals to maintain good health.</p>
</div>
</div>

```

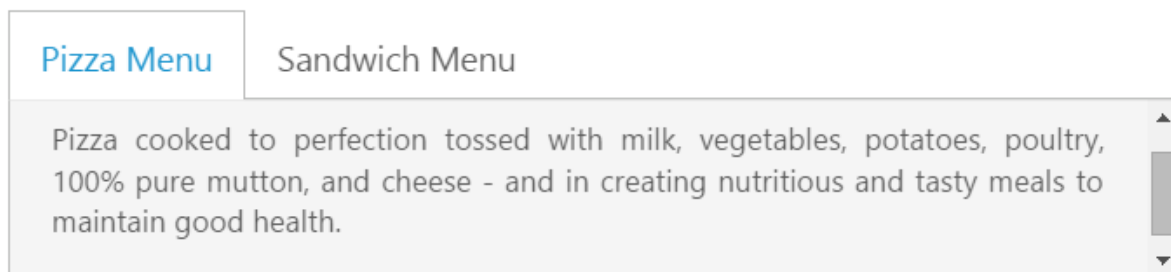
## JAVASCRIPT

```

// Add the following script to render Tab with customized height and
// height adjust mode.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
  $(function () {
    var sample = new ej.Tab($("#dish"), {
      heightAdjustMode: "fill",
      height: "300px"
    });
  });
}

```

The following screenshot illustrates the **Tab** with customized height and height adjust mode.



### Width

The **width** of the **Tab** widget is customized by using '**width**' property that accepts only the pixel values.

The following code example is used to render the **Tab** widget with customized width.

Add the following **HTML** to render **Tab** with customized width.

## HTML

```

<div id="dish" style="width: 650px">
  <ul>
    <li><a href="#pizza">Pizza Menu</a></li>
    <li><a href="#sandwich">Sandwich Menu</a></li>
  </ul>
  <div id="pizza" style="background-color: #F5F5F5">
    <!--Food item description-->
    <p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
    poultry, 100% pure mutton, and cheese - and in creating nutritious and
    tasty meals to maintain good health.</p>
  </div>

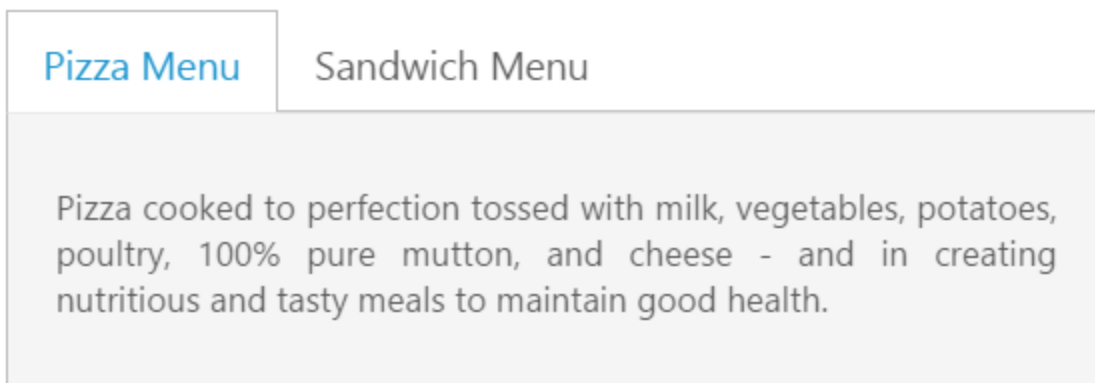
```

```
<div id="sandwich" style="background-color: #F5F5F5">
  <!--dish description-->
  <p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
  potatoes, poultry, 100% pure mutton, and cheese - and in creating
  nutritious and tasty meals to maintain good health.</p>
</div>
</div>
```

## JAVASCRIPT

```
// Add the following script to render Tab with customized width.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
  $(function () {
    var sample = new ej.Tab($("#dish"), {
      width: "450px"
    });
  });
}
```

The following screenshot illustrates the **Tab** with customized width.



## Theme

**Tab** control's style and appearance are controlled based on **CSS** classes. In order to apply styles to the **Tab** control, you can refer 2 files namely, **ej.widgets.core.min.css** and **ej.theme.min.css**. When the file **ej.widgets.all.min.css** is referred, then it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.widgets.all.min.css** is the combination of these two.

By default, there are 16 theme support available for RadialMenu component, namely:

- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark

- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap
- high-contrast-01
- high-contrast-02
- material
- office-365

### Custom styles

The style of the **Tab** widget is customized by 'cssClass' property.

The following code example is used to render the **Tab** widget with customized style.

Add the following **HTML** to render **Tab** with customized style.

#### HTML

```
<div id="dish" style="width: 650px">
<ul>
<li><a href="#pizza">Pizza Menu</a></li>
<li><a href="#sandwich">Sandwich Menu</a></li>
</ul>
<div id="pizza" style="background-color: #F5F5F5">
<!--Food item description-->
<p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>
<div id="sandwich" style="background-color: #F5F5F5">
<!--dish description-->
<p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
</div>
</div>
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
$(function () {
var sample = new ej.Tab($("#dish"), {
cssClass: "custom"
});
});
}
```

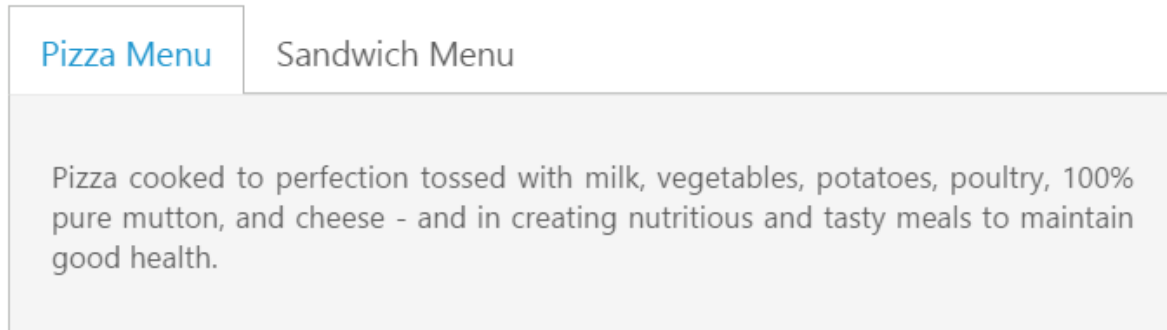
Add the following styles

#### CSS

```
<style type="text/css">
```

```
.custom {
width:650px;
}
</style>
```

The following screenshot illustrates the **Tab** with customized style.



### Integration with other widgets

You can provide more customization to the **Tab** with **rating** control as content in it for describing the item rating value.

The **Essential JavaScript Rating** control provide you an intuitive rating experience that allows you to select the number of stars that represents the rating. For more information about the rating, you can refer the following link:

<https://help.syncfusion.com/js/rating/getting-started>

The following code example explains you the **rating** control creation. The input element is used to create the **rating** control. Render the input element as **rating** control using the input element **id**. The code example is placed within the content description **<div>** element to declare the **rating** control and description in the **Tab** section and it is appended with the header initialization code section **<div>** element.

Add the following **HTML** to render **Tab** with other widget.

### HTML

```
<div id="dish" style="width: 650px">
<ul>
<li><a href="#pizza">Pizza Menu</a></li>
<li><a href="#sandwich">Sandwich Menu</a></li>
</ul>
<div id="pizza" style="background-color: #F5F5F5">
<p>Rating:</p>
<div class="dish">
<input id="pizza" type="text" class="rating" /><br />
</div>
<p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>
<div id="sandwich" style="background-color: #F5F5F5">
<p>Rating:</p>
<div class="dishRating">
```

```
<input id="sandwich" type="text" class="rating" />
</div>
<p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
</div>
</div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
$(function () {
var sample = new ej.Tab($("#dish"), {
});
var sample = new ej.Tab($("#pizza"), {
precision: ej.Rating.Precision.Exact,
value: 4.8
});
var sample = new ej.Tab($("#sandwich"), {
precision: ej.Rating.Precision.Exact,
value: 4.8
});
});
});
</script>
```

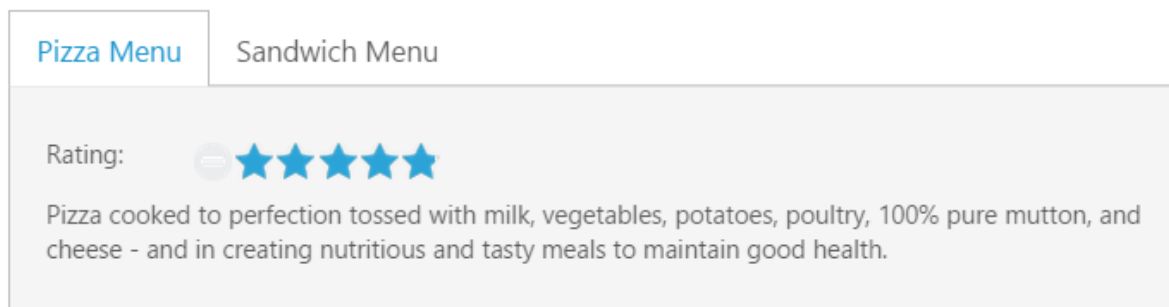
To render the **rating** controls in the first **Tab** element refer the styles mentioned in the following code example.

Add the following styles to render **Tab**.

## CSS

```
<style type="text/css" class="cssStyles">
.dishRating {
position: absolute;
margin: -31px 0px 0px 80px;
}
</style>
```

The following screenshot illustrates the **Tab** content with rating control.



## AJAX Content Load (Load on Demand)

You can change the contents in sub **Tab** element periodically and you are provided with a support to change the contents without any problems. To achieve the content load, use the **Load on Demand** concept.

In **Load On-Demand**, the external **HTML** file with the necessary details is referred in `<href>` section during **Tab** header declaration section. Also include `"dataType"`, `"contentType"`, and `"async"` in the script like the following example when rendering the control. When you click the **Tab** header, the AJAX automatically calls the content from the external files and displays in a **Tab** content section.

### Sub Tab with AJAX Content

Each item has a variety of options and these options are also specified in the limited space. So you can choose the **Tab** control that is used within the root **Tab** to specify more details.

The following code example illustrates to create the **Tab** control within the root **Tab** element. This **HTML** code is appended within the previous **HTML** code section. To render the child **Tab** with its header, add this code example within the first **Tab** `<div>` element.

Add the following **HTML** to render sub **Tab** with **AJAX** content.

### HTML

```
<div id="dish" style="width: 650px">
<ul>
<li><a href="#pizza">Pizza Menu</a></li>
<li><a href="#sandwich">Sandwich Menu</a></li>
</ul>
<div id="pizza" style="background-color: #F5F5F5">
<p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
<div id="pizza">
<ul>
<li>
<a href="content/cornSpinach.html">Corn & Spinach </a></li>
<li>
<a href="Content/chickenDelite.html">Chicken Delite </a></li>
</ul>
</div>
</div>
<div id="sandwich" style="background-color: #F5F5F5">
<p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
<div id="sandwich">
<ul>
<li>
<a href="Content/gardenVeggie.html">Garden Veggie </a></li>
<li>
<a href="Content/chickenTikka.html">Chicken Tikka </a></li>
<li>
<a href="Content/paneerTikka.html">Paneer Tikka </a></li>
</ul>
</div>
</div>
</div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
  $(function () {
    var sample = new ej.Tab($("#dish"), {
    });
    var sample1 = new ej.Tab($("#pizza"), {
      dataType: "html",
      contentType: "html",
      async: true
    });
    var sample2 = new ej.Tab($("#sandwich"), {
      dataType: "html",
      contentType: "html",
      async: true
    });
  });
}
```

The file 'cornSpinach.html' content is as follows.

## HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<div class="e-content">

<div class="ingredients">
Rate      : $70<br />
Ingredients : cheese, sweet corn &#38; green capsicums.
<br />
Description: Small pizza bases are topped with spinach and paneer and
fresh cream, a nice layer of mozzarella cheese. This is baked until the
cheese is all hot and gooey.
</div>
</div>
</body>
</html>
```

The file 'chickenDelite.html' content is as follows.

## HTML

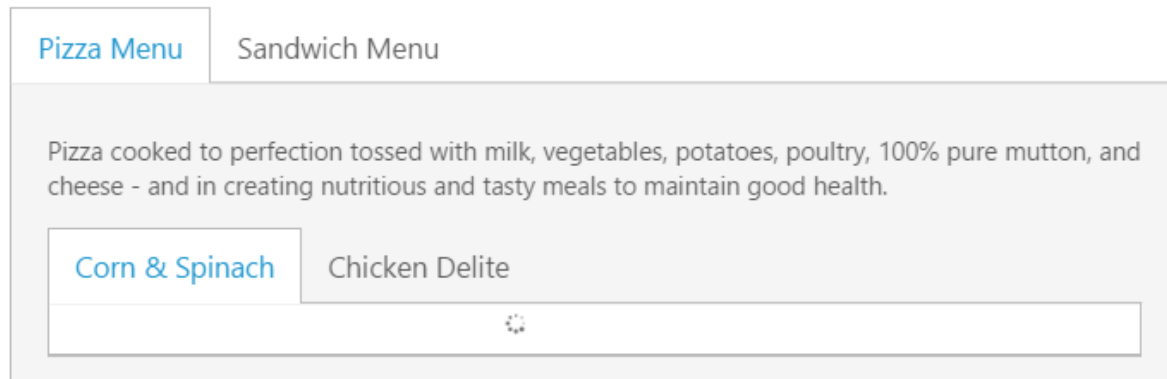
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<div class="e-content">

<div class="ingredients">
Rate      : $100<br />
```

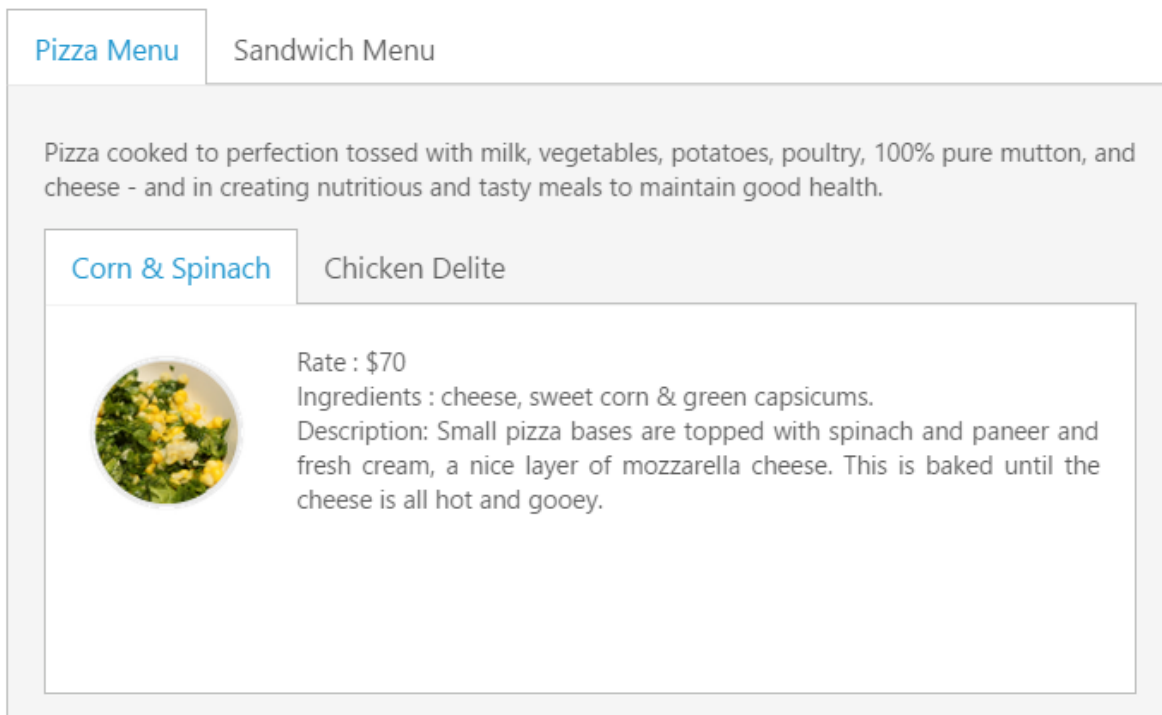


```
Ingredients : cheese, chicken chunks, onions &#38; pineapple chunks.  
<br />  
Description: This is a tasty, elegant chicken dish that is easy to  
prepare.  
</div>  
</div>  
</body>  
</html>
```

At the time of **Ajax** call, the content fetched from external file referenced location is illustrated in the following screenshot.



The following screenshot illustrates the First **Tab** with the sub **Tab** control using **Load on Demand**.



## RTL Support

**Tab** control provides support for load contents in right to left format. This is achieved by setting 'enableRTL' property to "true".

The following code example is used to render the **Tab** element in RTL format.

Add the following **HTML** to render **Tab** with **RTL** format.

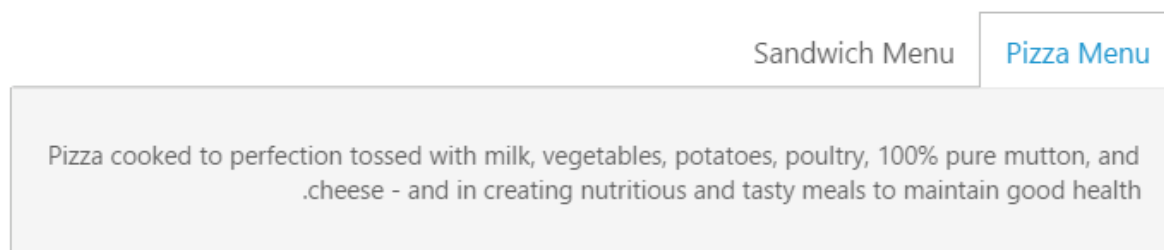
### HTML

```
<div id="dish" style="width: 650px">
<ul>
<li><a href="#pizza">Pizza Menu</a></li>
<li><a href="#sandwich">Sandwich Menu</a></li>
</ul>
<div id="pizza" style="background-color: #F5F5F5">
<!--Food item description-->
<p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>
<div id="sandwich" style="background-color: #F5F5F5">
<!--dish description-->
<p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
</div>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
$(function () {
var sample = new ej.Tab($("#dish"), {
enableRTL:true
});
});
}
```

The following screenshot illustrates the **Tab** with **RTL** format.



## Keyboard Navigation

**Tab** control provides supports **keyboard** interaction. Using this functionality you can interact with control using keyboard. This is achieved by enabling '**allowKeyboardNavigation**' to '**true**'. By default this property value is set to '**true**'.

Following table illustrates the accessible key and their usage

| Keys  | Behavior                |
|-------|-------------------------|
| Up    | Selected previous item. |
| Right | Selected previous item. |
| Down  | Selected next item.     |
| Left  | Selected next item.     |
| Home  | Selected first item.    |
| End   | Selected last item.     |

The following code example is used to render the **Tab** element in **RTL** format.

Add the following **HTML** to render **Tab** with **keyboard** navigation.

### HTML

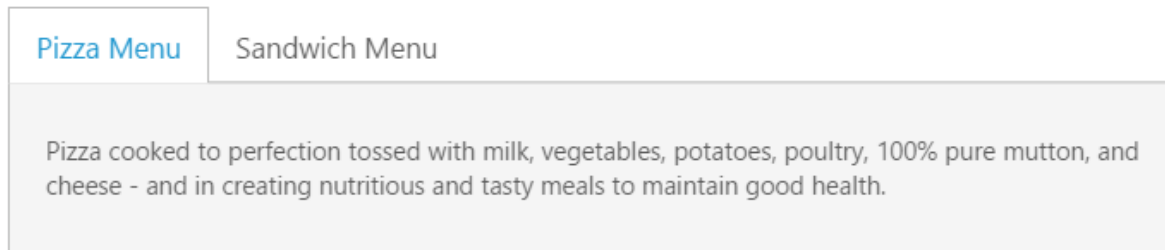
```
<div id="dish" style="width: 650px">
  <ul>
    <li><a href="#pizza">Pizza Menu</a></li>
    <li><a href="#sandwich">Sandwich Menu</a></li>
  </ul>
  <div id="pizza" style="background-color: #F5F5F5">
    <!--Food item description-->
    <p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
    poultry, 100% pure mutton, and cheese - and in creating nutritious and
    tasty meals to maintain good health.</p>
  </div>
  <div id="sandwich" style="background-color: #F5F5F5">
    <!--dish description-->
    <p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
    potatoes, poultry, 100% pure mutton, and cheese - and in creating
    nutritious and tasty meals to maintain good health.</p>
  </div>
</div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
  $(function () {
    var sample = new ej.Tab($("#dish"), {
      allowKeyboardNavigation: true
    });
  });
  //Control focus key
```

```
$(document).on("keydown", function (e) {
  if (e.altKey && e.keyCode === 74) {
    // j- key code.
    $("#dish ul a").focus();
  }
});
});
}
```

The following screenshot illustrates the **Tab** with **keyboard** navigation.



## Scroll Support

**Tab** control provides you scrolling support on Tab items to display a larger number of tabs with scroll buttons to get rid of the extending page size. The enabled Scroll buttons can be used to traverse through the elements.

By default, **Tab** header is rendered without scroll button. You can add the scroll button by setting the **"enableTabScroll"** property to **"true"**. When you move the cursor over the **Tab** header the scroll button is displayed.

You can use the following code example to render the **Tab** widget with scroll button.

Add the following **HTML** code to create a simple Tab with scroll button.

### HTML

```
<div style="width: 500px; padding: 50px;">
  <div id="dish">
    <ul>
      <li><a href="#pizza">Pizza Menu</a></li>
      <li><a href="#pasta">Pasta Menu</a></li>
      <li><a href="#burger">Burger Menu</a></li>
      <li><a href="#sandwich">Sandwich Menu</a></li>
      <li><a href="#spaghetti">Spaghetti Menu</a></li>
      <li><a href="#ramen">Ramen Menu</a></li>
    </ul>
    <div id="pizza" style="background-color: #F5F5F5">
      <!--Food item description-->
      <p>
        Pizza cooked to perfection tossed with milk, vegetables, potatoes,
        poultry, 100% pure mutton, and cheese - and in creating nutritious and
        tasty meals to maintain good health.</p>
    </div>
    <div id="pasta" style="background-color: #F5F5F5">
      <!--dish description-->
      <p>
```

```

Pasta cooked to perfection tossed with bread, milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>
<div id="sandwich" style="background-color: #F5F5F5">
<!--dish description-->
<p>
Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
</div>
<div id="burger" style="background-color: #F5F5F5">
<!--dish description-->
<p>
Burger cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
</div>
<div id="spaghetti" style="background-color: #F5F5F5">
<!--dish description-->
<p>
Spaghetti cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
</div>
<div id="ramen" style="background-color: #F5F5F5">
<!--dish description-->
<p>
Ramen cooked to perfection tossed with bread, milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>
</div>
</div>

```

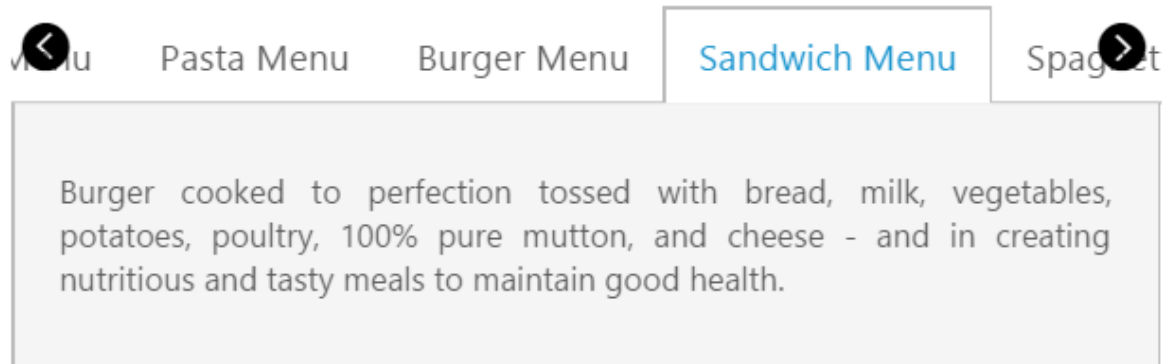
## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
$(function () {
var sample = new ej.Tab($("#dish"), {
enableTabScroll: true
});
});
}

```

The following screenshot illustrates you the **Tab** control with scroll button.



## Template Support

We can use this option to load any HTML elements and showcase it in the Tab panels as per our requirement.

We can load the contents or HTML elements directly inside the <div> element which we are going to convert as Tab control.

### HTML

```
<div id="dish" style="width: 650px">
<ul>
<li><a href="#corn">Corn & Spinach </a></li>
<li><a href="#chicken">Chicken Delite</a></li>
</ul>
<div id="corn" style="background-color: #F5F5F5">
<div class="e-content">

<div class="ingredients">
Rate      : $70<br /> Ingredients : cheese, sweet corn &#38; green
capsicums.
<br />
Description: Small pizza bases are topped with spinach and paneer and
fresh cream, a nice layer of mozzarella cheese. This is baked until the
cheese is all hot and gooey.
</div>
</div>
</div>
<div id="chicken" style="background-color: #F5F5F5">
<!--Content for Chicken Delite-->
</div>
</div>
```

### JAVASCRIPT


```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TabComponent {
$(function () {
var sample = new ej.Tab($("#dish"), {
});
```

```
}};  
}
```

**Output:**

Corn & Spinach

Chicken Delite



Rate : \$70  
Ingredients : cheese, sweet corn & green capsicums.  
Description: Small pizza bases are topped with spinach and paneer and fresh cream, a nice layer of mozzarella cheese. This is baked until the cheese is all hot and gooey.

## TagCloud

### Overview

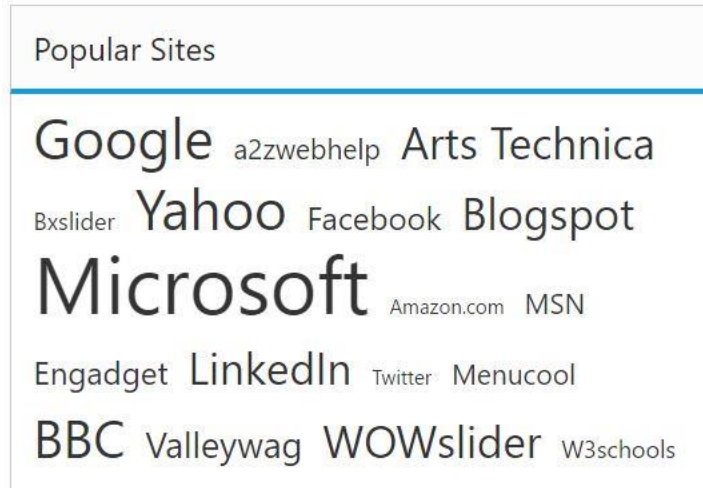
The JavaScript TagCloud control allows the user to display a list of links or tags with a structured cloud format where the importance of the tags can differentiate with varied font sizes, colors, and styles.

### Key Features

- **Data binding:** Supports data binding with JSON data as well as remote data.
- **Accessible:** Supports adding or removing tags dynamically.
- **Format:** Supports tag structure in both cloud format and list format.
- **Font-size:** Supports limiting the font size with a minimum and maximum range.
- **RTL:** Supports changing the TagCloud direction for right-to-left alignment.
- **Themes:** JavaScript controls include 12 built-in themes (6 flat and 6 gradient effects) and also support a custom skin option to set user defined themes.

### Getting Started

This section explains briefly about how to create a **TagCloud** in your application with **Typescript**. The **TagCloud** can be easily configured to the div element in which the tags are placed. The following screenshot illustrates the functionality of a **TagCloud** widget with a list of the topmost search engines.



Create TagCloud widget

Create an **HTML** page and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

Add necessary elements to render TagCloud

#### HTML

```
<div id="techWebList"></div>
```

Add the following code example to add list of items to the **TagCloud** and initialize the **TagCloud** widget.

#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TagCloudComponent {
var websiteCollection = [
{ text: "Google", url: "http://www.google.com", frequency: 12 },
{ text: "All Things Digital", url: "http://allthingsd.com/", frequency: 3
},
{ text: "Arts Technica", url: "http://arstechnica.com/", frequency: 8 },
```

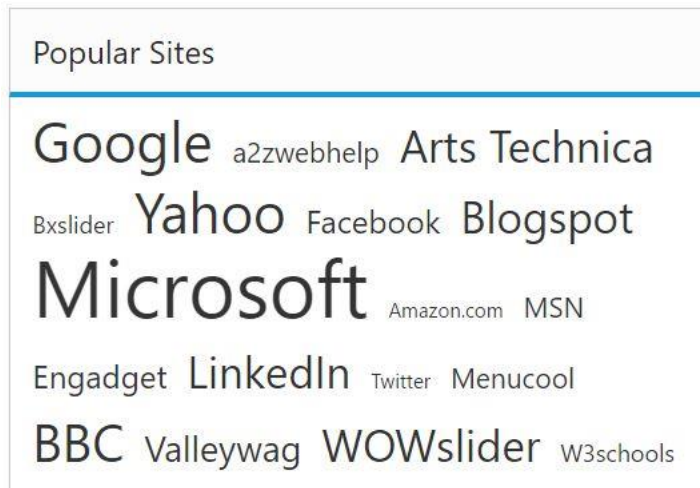


```

{ text: "Business Week", url: "http://www.businessweek.com/", frequency:
2 },
{ text: "Yahoo", url: "http://in.yahoo.com/", frequency: 12 },
{ text: "Center Networks", url: "http://www.centernetworks.com/",
frequency: 5 },
{ text: "Crave", url: "http://news.cnet.com/crave/", frequency: 8 },
{ text: "Crunch Gear", url: "http://techcrunch.com/gadgets/", frequency:
20 },
{ text: "Daily Tech", url: "http://www.dailytech.com/", frequency: 1 },
{ text: "Electronista", url: "http://www.electronista.com/", frequency: 3
},
{ text: "Engadget", url: "http://www.engadget.com/", frequency: 5 },
{ text: "Gearlog", url: "http://www.gearlog.com/", frequency: 9 },
{ text: "Information Week", url: "http://www.informationweek.com/",
frequency: 0 },
{ text: "PCWorld", url: "http://www.pcworld.com/", frequency: 11 },
{ text: "Tech Republic", url: "http://techrepublic.com/", frequency: 3 },
{ text: "Valleywag", url: "http://valleywag.gawker.com/", frequency: 6 },
{ text: "Rediff", url: "http://in.rediff.com/", frequency: 9 },
{ text: "WebProNews", url: "http://www.webpronews.com/", frequency: 2 }
];
$(function () {
var sample = new ej.TagCloud($("#techWebList"), {
titleText: "Popular sites",
dataSource: websiteCollection,
fields: {
text: "text", url: "url", frequency: "frequency"
}
});
});
}

```

The following screenshot displays the output of the above code example.



#### Set Min and Max Font Size

In the above code example, the **frequency** properties are used to set the min and max font size to the **TagCloud** list item.

#### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TagCloudComponent {
  var websiteCollection = [
    { text: "Google", url: "http://www.google.com", frequency: 12 },
    { text: "All Things Digital", url: "http://allthingsd.com/", frequency: 3
  },
    { text: "Arts Technica", url: "http://arstechnica.com/", frequency: 8 },
    { text: "Business Week", url: "http://www.businessweek.com/", frequency:
2 },
    { text: "Yahoo", url: "http://in.yahoo.com/", frequency: 12 },
    { text: "Center Networks", url: "http://www.centr networks.com/",
frequency: 5 },
    { text: "Crave", url: "http://news.cnet.com/crave/", frequency: 8 },
    { text: "Crunch Gear", url: "http://techcrunch.com/gadgets/", frequency:
20 },
    { text: "Daily Tech", url: "http://www.dailytech.com/", frequency: 1 },
    { text: "Electronista", url: "http://www.electronista.com/", frequency: 3
  },
    { text: "Engadget", url: "http://www.engadget.com/", frequency: 5 },
    { text: "Gearlog", url: "http://www.gearlog.com/", frequency: 9 },
    { text: "Information Week", url: "http://www.informationweek.com/",
frequency: 0 },
    { text: "PCWorld", url: "http://www.pcworld.com/", frequency: 11 },
    { text: "Tech Republic", url: "http://techrepublic.com/", frequency: 3 },
    { text: "Valleywag", url: "http://valleywag.gawker.com/", frequency: 6 },
    { text: "Rediff", url: "http://in.rediff.com/", frequency: 9 },
    { text: "WebProNews", url: "http://www.webpronews.com/", frequency: 2 }
  ];
  $(function () {
    var sample = new ej.TagCloud($("#techWebList"), {
      titleText: "Popular sites",
      dataSource: websiteCollection,
      fields: {
        text: "text", url: "url", frequency: "frequency"
      }
    });
  });
}

```

In the above code, the min font size is 0 and max font size is 12.

#### Set event to perform an operation

Here, you can set the **TagCloud** events such as create, mouseover, mouseout, click.

#### HTML

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TagCloudComponent {
  var websiteCollection = [
    { text: "Google", url: "http://www.google.com", frequency: 12 },
    { text: "All Things Digital", url: "http://allthingsd.com/", frequency: 3
  },
    { text: "Arts Technica", url: "http://arstechnica.com/", frequency: 8 },

```

```

{ text: "Business Week", url: "http://www.businessweek.com/", frequency:
2 },
{ text: "Yahoo", url: "http://in.yahoo.com/", frequency: 12 },
{ text: "Center Networks", url: "http://www.centernetworks.com/",
frequency: 5 },
{ text: "Crave", url: "http://news.cnet.com/crave/", frequency: 8 },
{ text: "Crunch Gear", url: "http://techcrunch.com/gadgets/", frequency:
20 },
{ text: "Daily Tech", url: "http://www.dailytech.com/", frequency: 1 },
{ text: "Electronista", url: "http://www.electronista.com/", frequency: 3
},
{ text: "Engadget", url: "http://www.engadget.com/", frequency: 5 },
{ text: "Gearlog", url: "http://www.gearlog.com/", frequency: 9 },
{ text: "Information Week", url: "http://www.informationweek.com/",
frequency: 0 },
{ text: "PCWorld", url: "http://www.pcworld.com/", frequency: 11 },
{ text: "Tech Republic", url: "http://techrepublic.com/", frequency: 3 },
{ text: "Valleywag", url: "http://valleywag.gawker.com/", frequency: 6 },
{ text: "Rediff", url: "http://in.rediff.com/", frequency: 9 },
{ text: "WebProNews", url: "http://www.webpronews.com/", frequency: 2 }
];
$(function () {
var sample = new ej.TagCloud($("#techWebList"), {
titleText: "Popular sites",
dataSource: websiteCollection,
fields: {
text: "text", url: "url", frequency: "frequency"
},
create: function(args) {
alert();
},
mouseover: function (args) {
alert();
},
mouseout: function (args) {
alert();
},
click: function (args) {
alert();
},
});
});
}

```

In the above code example, the **alert()** function is used to show the events that happened.

### Data-Binding

To render the **TagCloud** widget, it is necessary to bind the data to it in a proper way. The following sub-properties provides you a way to bind local or remote data to the **TagCloud** widget by binding the appropriate data fields to the corresponding options.

#### Fields

##### *dataSource*

This property assigns the local **JSON** data or remote (URL binding) data to the **TagCloud** control.

*query*

It accepts the data of object type that is usually the query string to fetch the required data from a specific table based on certain conditions. As this property is optional, when it is not specified, then the entire records that are initially assigned through dataSource is taken into consideration.

*text*

It maps the corresponding text field name from the data table or **JSON** data that is assigned to the dataSource with the text property of the **TagCloud** control. The text value that is fetched from the table renders the value to be displayed in the **TagCloud**.

*URL*

URL field in the data table or **JSON** data assigned the datasource is mapped to the URL property of the **TagCloud** control. The URL property defines the link to be navigated on clicking the corresponding text item.

*frequency*

It maps to the frequency field name from the data table or **JSON** data that is assigned to the dataSource. The frequency value that is fetched from the table should be a number to categorize the font size.

*Local Binding*

Local data binding allows you to map **JSON** data to **TagCloud**, that the corresponding text, URL, and frequency fields are assigned with a local **JSON** data.

*Defining the Local data for TagCloud*

The following steps explains you the local data binding to **TagCloud** widget,

In the **HTML** page, add a **<div>** element to configure **TagCloud** widget.

**HTML**

```
<div id="website"></div>
```

**JAVASCRIPT**

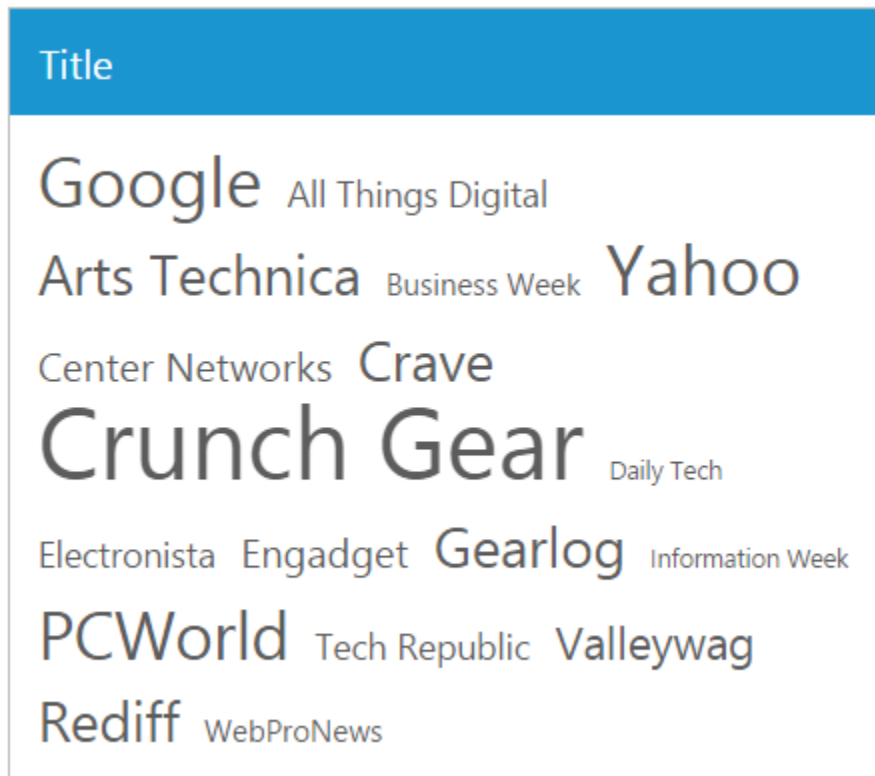
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TagCloudComponent {
  // Define local data source elements with text, url and frequency fields.
  var websiteCollection = [
    { text: "Google", url: "http://www.google.com", frequency: 12 },
    { text: "All Things Digital", url: "http://allthingsd.com/", frequency: 3 },
    { text: "Arts Technica", url: "http://arstechnica.com/", frequency: 8 },
    { text: "Business Week", url: "http://www.businessweek.com/", frequency: 2 },
    { text: "Yahoo", url: "http://in.yahoo.com/", frequency: 12 },
    { text: "Center Networks", url: "http://www.centernetworks.com/", frequency: 5 },
    { text: "Crave", url: "http://news.cnet.com/crave/", frequency: 8 },
    { text: "Crunch Gear", url: "http://techcrunch.com/gadgets/", frequency: 20 },
    { text: "Daily Tech", url: "http://www.dailytech.com/", frequency: 1 },
    { text: "Electronista", url: "http://www.electronista.com/", frequency: 3 },
    { text: "Engadget", url: "http://www.engadget.com/", frequency: 5 },
    { text: "Gearlog", url: "http://www.gearlog.com/", frequency: 9 },
```

```

{ text: "Information
Week",url:"http://www.informationweek.com/",frequency: 0 },
{ text: "PCWorld", url: "http://www.pcworld.com/", frequency: 11 },
{ text: "Tech Republic", url: "http://techrepublic.com/", frequency: 3 },
{ text: "Valleywag", url: "http://valleywag.gawker.com/", frequency: 6 },
{ text: "Rediff", url: "http://in.rediff.com/", frequency: 9 },
{ text: "WebProNews", url: "http://www.webpronews.com/", frequency: 2 }
];
$(function () {
var sample = new ej.TagCloud($("#website"), {
dataSource: websiteCollection
});
});
}

```

The following screenshot displays the **TagCloud** control with local data binding.



### Remote Binding

**TagCloud** provides remote data binding support to populate **TagCloud** items and the values can be mapped to the **TagCloud** fields from a remote web service by using **DataManager** and **Query**.

**DataManager** is used to manage relational data in **JavaScript**. It supports **CRUD** (Create, Read, Update, and Destroy) in individual requests and batch. **DataManager** use two different classes, **ej.DataManager** for processing, and **ej.Query** for serving data. **ej.DataManager** communicates with data source and **ej.Query** generates data queries that are read by **DataManager**.

### Configuring Remote data for TagCloud

The following steps explains you the local data binding to **TagCloud** widget.

In the **HTML** page, add a **<div>** element to configure **TagCloud** widget.

### HTML

```
<div id="website"></div>
```

Define **DataManager** and assign remote data source to it. Initialize query for binding data to **TagCloud** text. Here **DataManager** renders the remote web service and filters the data by using **Query**. The **select** property of **ejQuery**, is used to retrieve the specified columns from the data source.

### JAVASCRIPT

```
var dataManager = ej.DataManager({  
  url: "http://mvc.syncfusion.com/Services/Northwnd.svc/"  
});  
// Query creation  
var query = ej.Query().from("Orders").take(10);
```

Assign **datasource** and **query** property values to bind the remote data. Map the corresponding fields to **TagCloud** control as follows:

### JAVASCRIPT

```
var sample = new ej.TagCloud($("#website"), {  
  dataSource: dataManager,  
  query: query,  
  fields: { text: "CustomerID", frequency: "EmployeeID" }  
});
```

The following screenshot displays a **TagCloud** control with remote data binding.



## Title Customization

### Show title

The **TagCloud** items are displayed with a Title element by default. To hide the title, you can use **showTitle** property that is true by default.

### How to disable title in TagCloud

The following steps explain you on how to configure **title** for a **TagCloud**.

In the **HTML** page, add a **<div>** element to configure **TagCloud** widget.

## HTML

```
<div id="website"></div>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TagCloudComponent {
    // Disable the Title element of TagCloud control as follows.
    $(function () {
        var sample = new ej.TagCloud($("#website"), {
            showTitle: false,
            dataSource: websiteCollection
        });
    });
}
```

The following screenshot illustrates a **TagCloud** control when you disable title,



### Title text

**TagCloud** widget allows you to set a custom title text by using the **titleText** property. By default titleText property is set to string value "Title".

#### Defining title text for TagCloud

The following steps explain you on how to configure **titleText** for a **TagCloud**.

In the **HTML** page, add a **<div>** element to configure **TagCloud** widget.

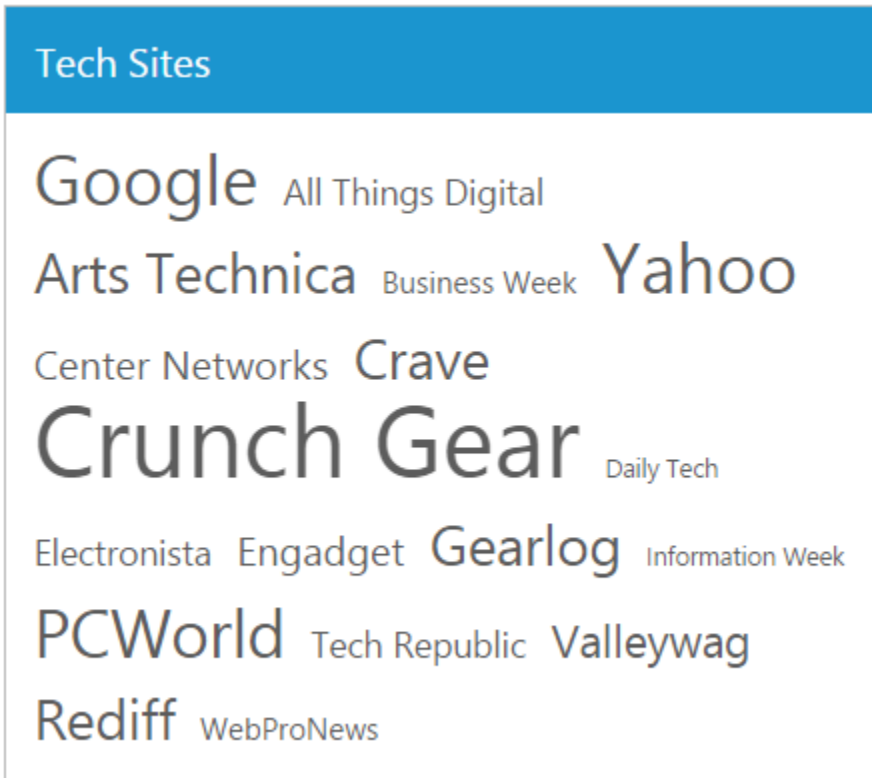
## HTML

```
<div id="website"></div>
```

## JAVASCRIPT

```
// Disable the Title element of TagCloud control as follows.  
var sample = new ej.TagCloud($("#website"), {  
  showTitle: true,  
  titleText: "Tech Sites",  
  dataSource: websiteCollection  
});
```

The following screenshot illustrates the **TagCloud** control with customized title text.



Title image

**TagCloud** widget provides **titleImage** to set an image for the title. You can set the desired image **URL** to **titleImage** property.

*Defining title text for TagCloud*

The following steps explain you to configure **titleImage** for a **TagCloud**.

In the **HTML** page, add a **<div>** element to configure **TagCloud** widget.

## HTML

```
<div id="website"></div>
```

## JAVASCRIPT

```
// Disable the Title element of TagCloud control as follows.  
var sample = new ej.TagCloud($("#website"), {
```



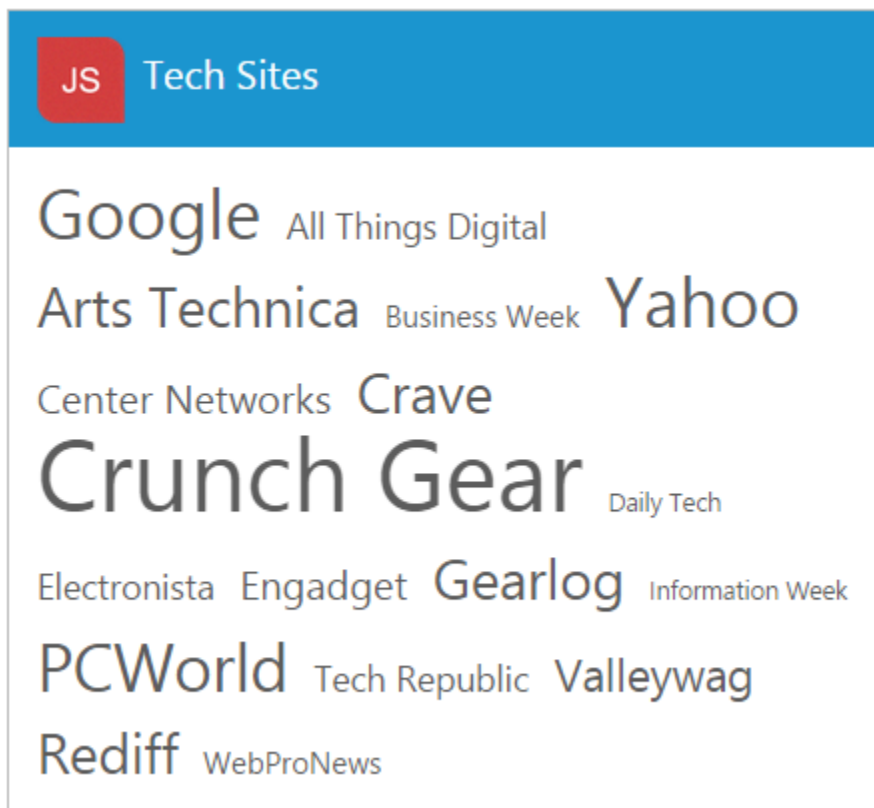
```
titleImage:  
"http://js.syncfusion.com/demos/web/images/waitingpopup/js_logo.png",  
titleText: "Tech Sites",  
dataSource: websiteCollection  
});
```

Using CSS class you can resize the image content as follows.

### CSS

```
<style type="text/css">  
.e-title-img {  
height:35px;  
width:35px;  
}  
</style>
```

The following screenshot illustrates the **TagCloud** control with customized title image.



### Appearance and Styling

#### Minimum and maximum Font size

The **TagCloud** content are set to different font sizes from minimum to maximum based on its frequency values. By default, **minFontSize** is "10px" and **maxFontSize** is "40px", using these properties you can customize the minimum and maximum font sizes.

### Customizing font sizes of TagCloud

The following steps explain you on how to configure font sizes for a **TagCloud**.

In the **HTML** page, add a **<div>** element to configure **TagCloud** widget.

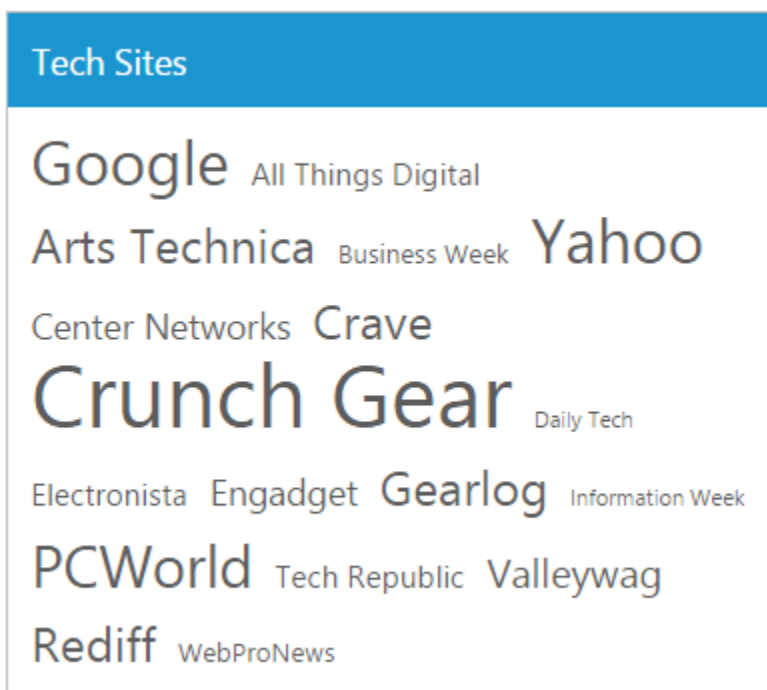
#### HTML

```
<div id="website"></div>
```

#### JAVASCRIPT

```
// Assign the values for minFontSize and maxFontSize properties below.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TagCloudComponent {
  $(function () {
    var sample = new ej.TagCloud($("#website"), {
      minFontSize: "20px",
      maxFontSize: "50px",
      titleText: "Tech Sites",
      dataSource: websiteCollection
    });
  });
}
```

The following screenshot illustrates the **TagCloud** control with customized font sizes.



#### Tag format

You can set the **TagCloud** content display format using **format** property. By default format is set to cloud, that displays content in **TagCloud**. You can set the format as list to display the content in linear format.

### Defining Cloud and List format

The following steps explain you to configure format for a **TagCloud**.

In the **HTML** page, add a **<div>** element to configure **TagCloud** widget.

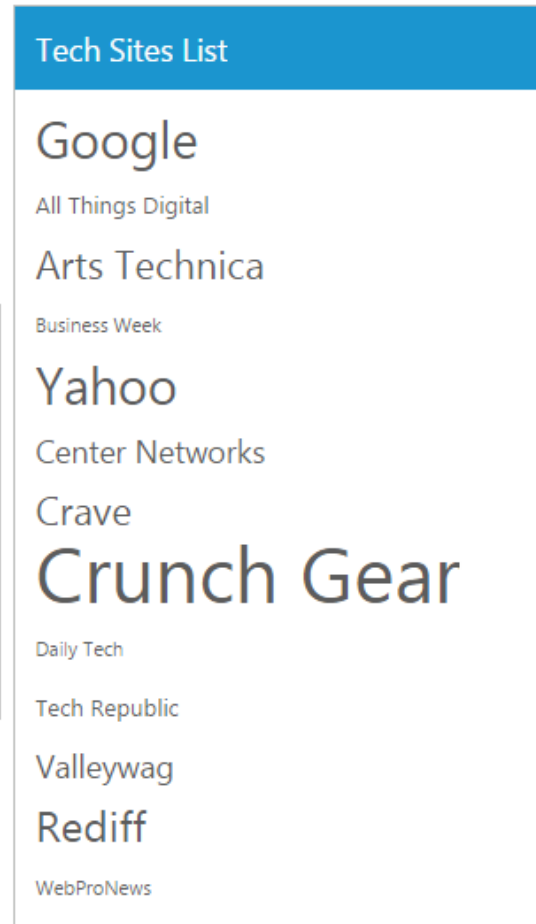
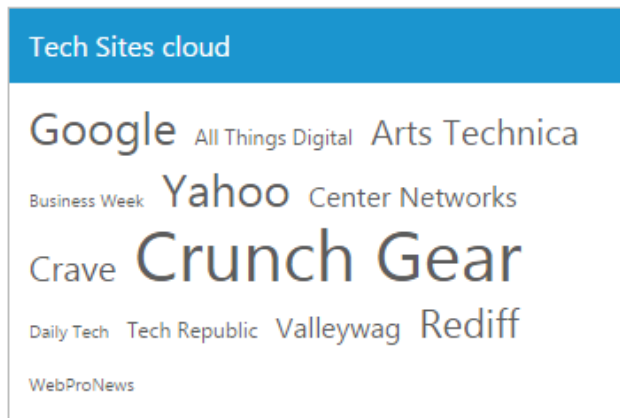
#### HTML

```
<table>
<tr>
<td>
<span>Tag format cloud</span>
<div id="techwebcloud"></div>
</td>
<td>
<span>Tag format list</span>
<div id="website"></div>
</td>
</tr>
</table>
```

#### JAVASCRIPT

```
// Assign the values for format property in TagCloud.
var sample = new ej.TagCloud($("#website"), {
  format: "list",
  titleText: "Tech Sites List",
  dataSource: websiteCollection
});
var sample = new ej.TagCloud($("#techwebcloud"), {
  format: "cloud",
  titleText: "Tech Sites Cloud",
  dataSource: websiteCollection
});
```

The following screenshot illustrates the **TagCloud** control with customized formats.



## Theme

You can control the style and appearance of **TagCloud** based on CSS classes. To apply styles to the **TagCloud** control, you can refer two files, **ej.widgets.core.min.css** and **ej.theme.min.css**. When you refer **ej.widgets.all.min.css** file, it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.widgets.all.min.css** is the combination of these two.

By default, there are 16 theme support available for RadialMenu component, namely:

- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap

- high-contrast-01
- high-contrast-02
- material
- office-365

### CssClass

You can use the CSS class to customize the **TagCloud** appearance. Any of the CSS properties can be used to modify look and feel of tag cloud based on the requirement. Define a **CSS** class as per requirement and assign the class name to **cssClass** property.

#### Configure TagCloud using CSS class

The following steps allows you to configure **CSS** class for **TagCloud**.

In the **HTML** page, add a **<div>** element to configure **TagCloud** widget.

#### HTML

```
<div id="website"></div>
```

#### JAVASCRIPT

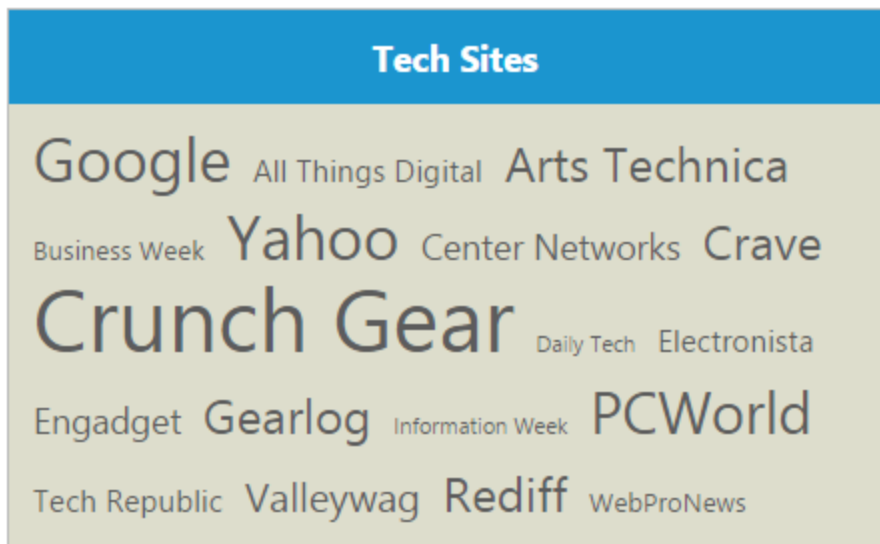
```
// Add the cssClass property to TagCloud.
var sample = new ej.TagCloud($("#website"), {
  titleText: "Tech Sites",
  dataSource: websiteCollection,
  cssClass: "CustomCss"
});
```

Define CSS class for customizing the **TagCloud** widget.

#### CSS

```
<style type="text/css" class="cssStyles">
/* Customize the TagCloud div element */
.CustomCss
{
background-color: #DDC;
width: 400px;
}
/* Customize the TagCloud header element */
.CustomCss .e-header.e-title {
text-align: center;
font-weight: bold;
}
</style>
```

The following screenshot illustrates the **TagCloud** with customized CSS class,



### RTL Support

This feature supports you to change the left-to-right alignment of the **TagCloud** widget to right-to-left (RTL). This displays the content from right-to-left in the widget. You can achieve this using **enableRTL** property that is set false by default.

### Enabling RTL Support

The following steps explains you the enabling of right-to-left property in **TagCloud**.

In the **HTML** page, add a **<div>** element to configure **TagCloud** widget.

#### HTML

```
<div id="website"></div>
```

#### JAVASCRIPT

```
// Enable RTL property for TagCloud.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TagCloudComponent {
    $(function () {
        var sample = new ej.TagCloud($("#website"), {
            enableRTL:true,
            titleText: "Tech Sites",
            dataSource: websiteCollection
        });
    });
}
```

The following screenshot illustrates the **TagCloud** control with RTL support.



## Tile

### Overview

The Typescript Tile control displays an opaque rectangles or squares and they are arrayed on the start screen like a grid pattern. Tapping or selecting a Tile, launches the app or does some other action that is represented by the Tile. Tiles are arranged in a group separated by columns that looks like a start screen of a device and it can be either static or live.

### Key Features

- **Live tile support:** Here the tiles are changed dynamically at specific time interval. *Badge value:* It is used to show the content count present behind the Tile. **Caption support:** Caption is used to set title for the corresponding tile.

### Getting Started

This section helps you to understand the getting started of the Tile control with the step-by-step instructions.

#### Create a Tile

The following steps guide you to add a Tile control.

1) Refer the common Typescript [Getting Started Documentation](#) to create an application and add necessary scripts and styles for rendering the Tile control. 2) Create an HTML page and add the scripts and css references in the order mentioned in the following code example.

#### HTML

```

<!DOCTYPE html>
<html>
<head>
<title>Typescript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/fl
at-azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.
web.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add Tile here-->
</body>
</html>

```

3) Add the below mentioned code with in the tag to render the Tile control with tile text, size and image.

#### HTML

```

<div class="e-tile-column">
<div id="tile1"></div>
</div>

```

Initialize the Tile in ts file by using the ej.Tile method.

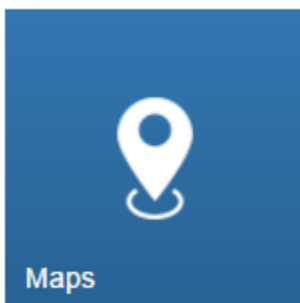
#### TS

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module TileComponent {
$(function () {
var tile1 = new ej.Tile($("#tile1"), {
tileSize:"medium",
imageUrl:'http://js.syncfusion.com/ug/web/content/tile/map.png',
caption:{text:"Maps"}
});
});
}

```

Get the following output from the above mentioned code





You can easily design a home page using tile control for easy navigation. Therefore, you require many different sizes of tiles aligned in a grid-like manner. The tiles will be aligned automatically to define the necessary tile elements inside the wrapper element, that contains a column class. You can define all columns elements under the wrapper element with the tile group class to make 'n' number of tiles as a grouped tile. Add the below mentioned code to the corresponding view page.

### HTML

```
<div id="scrollTarget">
<div class="e-tile-group">
<div class="e-tile-column">
<div id="tile1"></div>
<div class="e-tile-small-col-2">
<div id="tile2"></div>
<div id="tile3"></div>
<div id="tile4"></div>
<div id="tile5"></div>
</div>
<div id="tile6"></div>
<div id="tile7"></div>
<div id="tile8"></div>
</div>
<div class="e-tile-column">
<div id="tile9"></div>
<div id="tile10"></div>
<div id="tile11"></div>
<div id="tile12"></div>
<div id="tile13"></div>
</div>
</div>
</div>
```

### TS

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module TileComponent {
$(function () {
var tile1 = new ej.Tile($("#tile1"), {
imagePosition:"fill",
caption:{text:"People"},
tileSize:"medium",
imageUrl: 'http://js.syncfusion.com/ug/web/content/tile/people_1.png',
});
var tile2 = new ej.Tile($("#tile2"), {
imagePosition:"center",
tileSize:"small",
imageUrl:'http://js.syncfusion.com/ug/web/content/tile/alerts.png',
});
var tile3 = new ej.Tile($("#tile3"), {
imagePosition:"center",
tileSize:"small",
imageUrl:'http://js.syncfusion.com/ug/web/content/tile/bing.png',
});
var tile4 = new ej.Tile($("#tile4"), {
tileSize:"small",
```

```
imageUrl:'http://js.syncfusion.com/ug/web/content/tile/camera.png',
});
var tile5 = new ej.Tile($("#tile5"), {
  imagePosition:"center",
  tileSize:"small",
  imageUrl:'http://js.syncfusion.com/ug/web/content/tile/messages.png',
});
var tile6 = new ej.Tile($("#tile6"), {
  imagePosition:"center",
  tileSize:"medium",
  imageUrl:'http://js.syncfusion.com/ug/web/content/tile/games.png',
  caption:{text:"Play"}
});
var tile7 = new ej.Tile($("#tile7"), {
  tileSize:"medium",
  imageUrl:'http://js.syncfusion.com/ug/web/content/tile/map.png',
  caption:{text:"Maps"}
});
var tile8 = new ej.Tile($("#tile8"), {
  imagePosition:"fill",
  tileSize:"wide",
  imageUrl:'http://js.syncfusion.com/ug/web/content/tile/sports.png',
  caption:{text:"Sports"}
});
var tile9 = new ej.Tile($("#tile9"), {
  imagePosition:"fill",
  tileSize:"medium",
  imageUrl:'http://js.syncfusion.com/ug/web/content/tile/people_2.png',
  caption:{text:"People"}
});
var tile10 = new ej.Tile($("#tile10"), {
  imagePosition:"center",
  tileSize:"medium",
  imageUrl:'http://js.syncfusion.com/ug/web/content/tile/pictures.png',
  caption:{text:"Photo"}
});
var tile11 = new ej.Tile($("#tile11"), {
  imagePosition:"center",
  tileSize:"wide",
  imageUrl:'http://js.syncfusion.com/ug/web/content/tile/weather.png',
  caption:{text:"Weather"}
});
var tile12 = new ej.Tile($("#tile12"), {
  imagePosition:"center",
  tileSize:"medium",
  imageUrl:'http://js.syncfusion.com/ug/web/content/tile/music.png',
  caption:{text:"Music"}
});
var tile13 = new ej.Tile($("#tile13"), {
  imagePosition:"center",
  tileSize:"medium",
  imageUrl:'http://js.syncfusion.com/ug/web/content/tile/favs.png',
  caption:{text:"Favorites"}
});
});
}
```

Run the above code to get the following output.



---

**Note:** You can find the Tile control properties from the [API reference](#).

---

### Image Configuration

The **“data-ej-imagePosition”** attribute is used to adjust the position of **Tile** image at the **center** on initialization. The possible values for the **“data-ej-imagePosition”** are as follows

1. Center
2. Top
3. Bottom
4. Right
5. Left
6. TopLeft
7. BottomRight
8. BottomLeft
9. Fill

The **“data-ej-imageUrl”** attribute is used to set the background image for **Tile**, where the image is given in the path specified by **“data-ej-imageUrl”** attribute.

Refer to the following code examples.

## HTML

```
<div id="tile"></div>
```

Add the following code inside the **script** tag.

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TileViewComponent {
    $(function () {
        var tile1 = new ej.Tile($("#tile1"), {
            tileSize: "wide",
            imagePosition: "center",
            imageUrl: "http://js.syncfusion.com/UG/web/Content/tile/Weather_2.png",
            text: "weather"
        });
    });
}
```



You can give images for each tile through **css** classes by using "**data-ej-imageClass**" attribute. You can define your desired styles in the specified class.

Refer to the following code examples.

## HTML

```
<div id="tile"></div>
```

## CSS

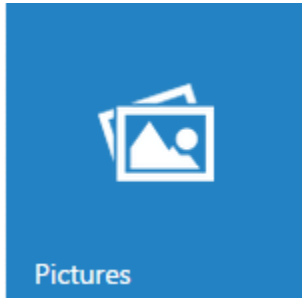
```
<style>
.pictures {
background:
url("http://js.syncfusion.com/UG/web/Content/tile/pictures.png");
background-size: 30px 30px;
}
</style>
```

Add the following code inside the **script** tag.

## JAVASCRIPT

```
var tile1 = new ej.Tile($("#tile1"), {
```

```
tileSize: "medium",  
imagePosition: "center",  
imageClass: "pictures",  
text: "Pictures"  
});
```



### Text Configuration

The **"data-ej-showText"** attribute is used to show or hide the **Tile** caption. By default, the **"data-ej-showText"** attribute is to **true** on initialization. The **"data-ej-text"** attribute is used to set the caption of a **Tile** as a **Text** on initialization. The **"data-ej-textAlignment"** attribute is used to align the **Tile** text as **normal** on initialization. The possible position values for **"data-ej-textAlignment"** are as follows:

1. normal
2. left
3. right
4. center

Refer to the following code examples.

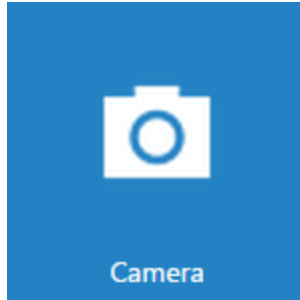
### HTML

```
<div id="tile"></div>
```

Add the following code inside the **script** tag.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module TileViewComponent {  
    $(function () {  
        var tile1 = new ej.Tile($("#tile1"), {  
            tileSize: "medium",  
            imagePosition: "center",  
            textAlignment: "center",  
            imageUrl: "http://js.syncfusion.com/UG/web/Content/tile/camera.png",  
            text: "Camera"  
        });  
    });  
}
```



## Template Support

The “**data-ej-imageTemplateId**” attribute is used to customize the image of **Tile** with template feature by setting the **id**. The “**data-ej-captionTemplateId**” attribute is used to customize the text of **Tile** with template feature by setting the **id**.

Refer to the following code examples.

### HTML

```
<div id="tile"></div>
<div id="imageTemplate">
<div id="appimage">
</div>
<div class="tileMargin">
<span class="caption">Google Search</span><br />
<span class="description">The world's information</span><br />
<span class="sub">Free</span>
</div>
</div>
<div id="captionTemplate" class="title">Windows Store</div>
```

Add the following code inside the **script** tag.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TileViewComponent {
$(function () {
var tile1 = new ej.Tile($("#tile1"), {
tileSize: "wide",
imageTemplateId: "imageTemplate",
captionTemplateId: "captionTemplate"
});
});
}
```

Add the following code into sample.

### CSS

```
<style>
#appimage {
background-image:
url("http://js.syncfusion.com/UG/mobile/content/google.png");
background-position: center center;
```

```
background-repeat: no-repeat;
background-size: 50% auto;
display: table-cell;
width: 45%;
}
.tileMargin {
display: table-cell;
padding-top: 25px;
}
.e-tile-template {
display: table;
height: 100%;
width: 100%;
}
</style>
```



### Configure Badge

The **badge** property handles badge specific functionalities like enable or disable the badge and setting badge value for **Tile**.

The “**data-ej-badge-enabled**” attribute enables or disables the badge for a **Tile**. The **Tile** renders with hidden badge when it is set to **false**.

The “**data-ej-badge-value**” attribute is used to set the badge value to a **Tile**. By default, the **Value** is set to **1** on initialization. The “**data-ej-badge-text**” attribute is used to set the text instead of number for **Tile** badge.

The “**data-ej-badge-maxValue**” attribute is used to set the maximum badge value to a **Tile**. When you set the badge value greater than “**data-ej-badge-maxValue**”, it shows maximum value in badge with **plus** symbol.

The “**data-ej-badge-minValue**” attribute is used to set the minimum badge value to a **Tile**. When you set the badge value less than “**data-ej-badge-minValue**”, it shows minimum value in badge.

Refer to the following code examples.

#### HTML

```
<div id="tile"></div>
```

Add the following code inside the **script** tag.

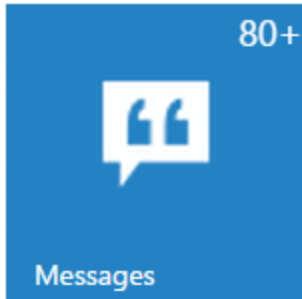
#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

```

module TileViewComponent {
  $(function () {
    var tile1 = new ej.Tile($("#tile1"), {
      tileSize: "medium", imagePosition: "center",
      badge: { enabled: true, minValue: 10, maxValue: 80, value: 88 },
      text: "Messages", imageUrl:
        "http://js.syncfusion.com/UG/Web/Content/tile/messages.png"
    });
  });
}

```



### LiveTile Configuration

**Live Tiles** are used to display the current or up to date information like scores, stocks, weather, etc. You can enable **Live Tile** using `[data-ej-livetile-enabled]` attribute by setting it to **true**. The `[data-ej-livetile-type]` attribute allows you to specify the type of animation while updating the information in **Tile**. There are three types of **Tile** animation supported: Flip, Slide and Carousel.

The `[data-ej-livetile-imageUrl]` attribute sets background image for **Live Tile**. This property accepts array values so you can specify the image **url's** for all the **Tiles** that are used in single **Live Tile**.

You can specify time interval for each **Tile** update/animation using `[data-ej-livetile-updateInterval]` attribute. Time interval is given in milliseconds. The default value is 2000.

Refer to the following code examples.

### HTML

```
<div id="tile"></div>
```

Add the following code inside the **script** tag.

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TileViewComponent {
  $(function () {
    var tile1 = new ej.Tile($("#tile1"), {
      tileSize: "medium", imagePosition: "fill",
      liveTile: {
        updateInterval: 2500, type: "flip", enabled: true,
        imageUrl: ['http://js.syncfusion.com/UG/Web/Content/tile/people_1.png',
          'http://js.syncfusion.com/UG/Web/Content/tile/people_2.png']
      },
      text: "Peoples"
    });
  });
}

```



```

    });
  });
}

```

In `[data-ej-livetile-imageTemplateId]` attribute, you can give **Live Tile** images outside the **Tile** rendering. To achieve this, you are required to give image content inside the element where the path is specified by `templateId`. You can update the “`imageTemplateId`” dynamically through `updateTemplateId` public method.

Refer to the following code examples.

### HTML

```

<div id="tile"></div>
<div id="temp1" style="background-image:
url('http://js.syncfusion.com/UG/Web/Content/tile/people_1.png'); width:
100%; height: 100%;">
</div>
<div id="temp2" style="background-image:
url('http://js.syncfusion.com/UG/Web/Content/tile/people_2.png'); width:
100%; height: 100%;">
</div>

```

Add the following code inside the **script** tag.

### JAVASCRIPT

```

var tile1 = new ej.Tile($("#tile1"), {
  tileSize: "medium", imagePosition: "fill",
  liveTile: {
    updateInterval: 2500, type: "flip", enabled: true,
    imageTemplateId: ["temp1", "temp2"]
  },
  text: "Peoples"
});

```

You can specify the array of images for **Live Tile** through **CSS** classes by using `[data-ej-livetile-imageClass]` attribute and you can define the desired styles in the specified class.

Refer to the following code examples.

### HTML

```

<div id="tile"></div>

```

### CSS

```

<style>
.people1 {
background-image:
url('http://js.syncfusion.com/UG/Web/Content/tile/people_1.png');
width: 100%;
height: 100%;
}
.people2 {

```

```
background-image:
url('http://js.syncfusion.com/UG/Web/Content/tile/people_2.png');
width: 100%;
height: 100%;
}
</style>
```

Add the following code inside the **script** tag.

#### JAVASCRIPT

```
var tile1 = new ej.Tile($("#tile1"), {
  tileSize: "medium", imagePosition: "fill",
  liveTile: {
    updateInterval: 2500, type: "flip", enabled: true,
    imageClass: ["people1", "people2"]
  },
  text: "Peoples"
});
```

#### Customize size

You can customize the size of the **Tile** by using “**data-ej-tileSize**” attribute. The following built-in tile sizes are supported.

1. medium
2. small
3. large
4. wide

The default **TileSize** value is set to **small**.

Refer to the following code examples.

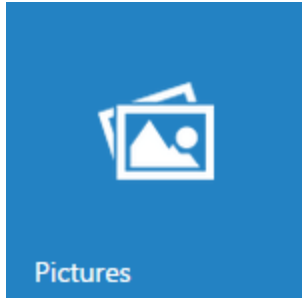
#### HTML

```
<div id="tile"></div>
```

Add the following code inside the **script** tag.

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TileViewComponent {
  $(function () {
    var tile1 = new ej.Tile($("#tile1"), {
      tileSize: "medium", imagePosition: "center",
      imageUrl: "http://js.syncfusion.com/UG/Web/Content/tile/pictures.png",
      text: "Pictures"
    });
  });
}
```



### Add Group Tiles

To make a **Tile** as grouped tile, you can use the following mentioned pre-defined classes.

| Class Name  | Explanation                        |
|-------------|------------------------------------|
| group       | To group the column elements       |
| column      | To align the tile in column manner |
| small-col-2 | To align the small size tiles      |

To render group tile, refer to the following code example.

#### HTML

```
<div class="**group**">
  <div class="**column**">
    <!-- Add tile control here -->
  </div>
</div>
```

To render **column** grouped tile, you need to render the number of tiles inside a **<div>** element with class **'column'**. Then that column group element is appended to a **<div>** with class **'group'**.

To render **small-col-2** grouped tile, you need to render the number of tiles inside a **<div>** element with class **'small-col-2'**. Then that **small-col-2** group element is appended to a **<div>** with class **'column'**. Then you need to append those column inside the main group **<div>** element.

Refer the following code examples.

#### HTML

```
<div class="group">
  <div class="column">
    <div id="tile1">
    </div>
    <div id="tile2">
    </div>
    <div id="tile3">
    </div>
    <div id="tile4">
    </div>
    <div id="tile5">
    </div>
  </div>
</div>
```

```
<div class="column">
<div id="tile6">
</div>
<div id="tile7">
</div>
<div id="tile8">
</div>
<div id="tile9">
</div>
<div id="tile10">
</div>
<div id="tile11">
</div>
</div>
</div>
```

Add the following code inside the **script** tag.

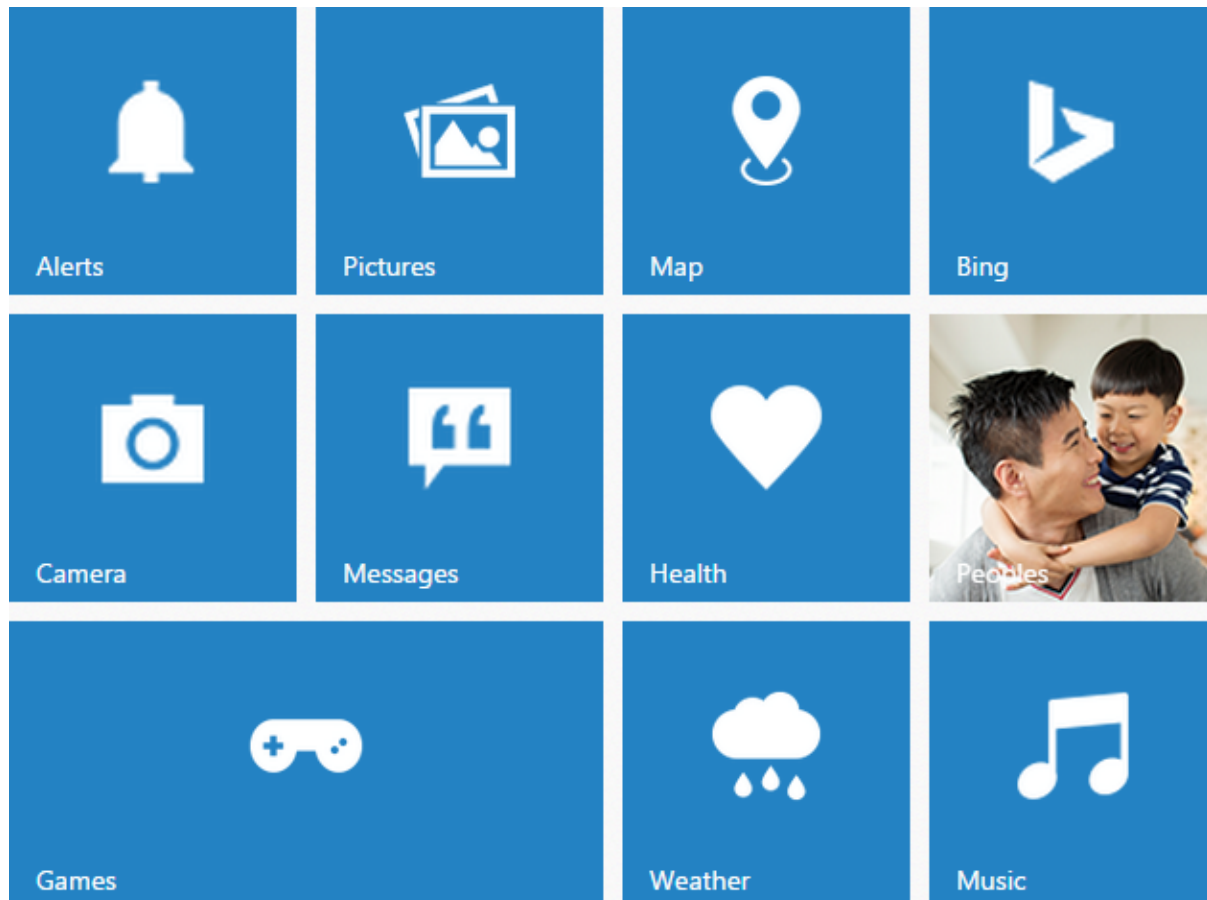
### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TileViewComponent {
$(function () {
var tile1 = new ej.Tile($("#tile1"), {
tileSize: "medium",
imageUrl: "http://js.syncfusion.com/UG/Web/Content/tile/alerts.png",
text: "Alerts"
});
var tile2 = new ej.Tile($("#tile2"), {
tileSize: "medium",
imageUrl: "http://js.syncfusion.com/UG/Web/Content/tile/pictures.png",
text: "Pictures"
});
var tile3 = new ej.Tile($("#tile3"), {
tileSize: "medium",
imageUrl: "http://js.syncfusion.com/UG/Web/Content/tile/camera.png",
text: "Camera"
});
var tile4 = new ej.Tile($("#tile4"), {
tileSize: "medium",
imageUrl: "http://js.syncfusion.com/UG/Web/Content/tile/messages.png",
text: "Messages"
});
var tile5 = new ej.Tile($("#tile5"), {
tileSize: "wide",
imageUrl: "http://js.syncfusion.com/UG/Web/Content/tile/games.png",
text: "Games"
});
var tile6 = new ej.Tile($("#tile6"), {
tileSize: "medium",
imageUrl: "http://js.syncfusion.com/UG/Web/Content/tile/map.png",
text: "Map"
});
var tile7 = new ej.Tile($("#tile7"), {
tileSize: "medium",
```

```

imageUrl: "http://js.syncfusion.com/UG/Web/Content/tile/bing.png",
text: "Bing"
});
var tile8 = new ej.Tile($("#tile8"), {
tileSize: "medium",
imageUrl: "http://js.syncfusion.com/UG/Web/Content/tile/favs.png",
text: "Health"
});
var tile9 = new ej.Tile($("#tile9"), {
tileSize: "medium", imagePosition: "fill",
imageUrl: "http://js.syncfusion.com/UG/Web/Content/tile/people_2.png",
text: "Peoples"
});
var tile9 = new ej.Tile($("#tile10"), {
tileSize: "medium",
imageUrl: "http://js.syncfusion.com/UG/Web/Content/tile/weather.png",
text: "Weather"
});
var tile9 = new ej.Tile($("#tile11"), {
tileSize: "medium",
imageUrl: "http://js.syncfusion.com/UG/Web/Content/tile/music.png",
text: "Music"
});
});
}

```



## TimePicker

### Overview

**TimePicker** control allows you to select the time value with a specific format.

### Key Features

- Time format: Supports all valid time formats.
- Localization: Supports localization to different cultures.
- Persist: Supports state maintenance during page refresh.
- RTL: Support for Right to Left alignment of content in TimePicker control.
- Themes: Essential JavaScript controls feature 12 built-in themes (six flat themes and six with gradient effects), and also supports custom skin options to set user-defined themes.

### Getting Started

This section explains you how to render and configure TimePicker component in a TypeScript application.

#### Create your first TimePicker

1. Create a TypeScript application and refer the dependent modules, script and CSS with the help of given Getting started document.
2. In the index.HTML file, add the input element for rendering TimePicker component as given below.

#### HTML

```
<input id="timepick" />
```

3. Create a TypeScript file named "app.ts" file and refer the required definition files as given below.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />
```

4. Now, initialize the TimePicker component by using ej.TimePicker method.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
$(function () {  
  var sample = new ej.TimePicker($("#timepick"));  
});
```

### Run the application

To run the application, navigate the project folder and open command prompt window and execute the following command.

#### JS

```
tsc
```

This command compiles the app.ts file to generate a JS file named app.js file.

Refer the app.js file in index.html and browse the html file to see the following output.



### Configuring Properties

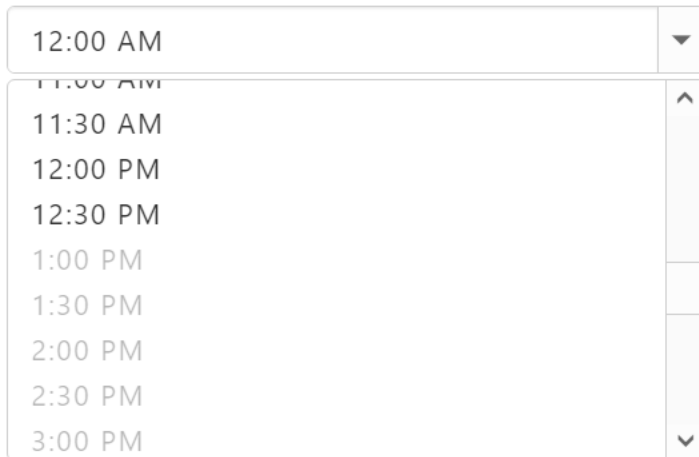
#### DisableTimeRanges

This property specifies the list of time range to be disabled in TimePicker control. To know more about TimePicker properties please refer the [API reference] (<https://help.syncfusion.com/api/js/ejtimepicker>) documentation.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TimePickerComponent {
  $(function () {
    var sample = new ej.TimePicker($("#timepick"), { disableTimeRanges: [{
      startTime: "3:00 AM", endTime: "6:00 AM" },
      { startTime: "1:00 PM", endTime: "3:00 PM" },
      { startTime: "8:00 PM", endTime: "10:00 PM" }]});
  });
}
```

The following screenshot illustrates the output of above code.



## Behavior Settings

Set value of the TimePicker widget

You can use value property to set default time for the **TimePicker**.

The following steps explain you on how to set the default value of the **TimePicker**.

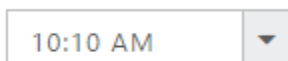
In the **HTML** page, add a **<input>** element to configure **TimePicker** widget.

### HTML

```
<input type="text" id="time" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TimePickerComponent {
    // Set default value of the TimePicker as follows.
    $(function () {
        var timeSample = new ej.TimePicker($("#timepick"), {
            value: "10:10 AM"
        });
    });
}
```



## Enable/Disable TimePicker widget

**TimePicker** widget provides you an option to enable /disable the widget. You can disable the TimePicker by setting the “**enabled**” property value as **false**.



The following steps explain you to enable/disable property in **TimePicker** widget.

In the **HTML** page, add a **<input>** element to configure **TimePicker** widget.

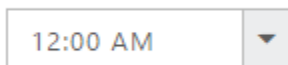
### HTML

```
<input type="text" id="time" />
```

### JAVASCRIPT

```
// To enable/disable TimePicker controls use the following code example.  
//Enable Timepicker:  
$(function () {  
    var timeSample = new ej.TimePicker($("#timepick"), {  
    });  
    timeSample.enable();  
});  
//Disable Timepicker:  
$(function () {  
    var timeSample = new ej.TimePicker($("#timepick"), {  
    });  
    timeSample.disable();  
});
```

Execute the above code to render the following output.



### Restrict editing

**TimePicker** widget provides **readOnly** property to disable editing in the control. Therefore you can only read the value set to **TimePicker** and cannot modify it. The **value** property allows you to set the default value for **TimePicker** widget when it is created.

#### Configure TimePicker textbox to restrict editing

The following steps allows you to disable editing value in **TimePicker**.

In the **HTML** page, add a **<input>** element to configure **TimePicker** widget.

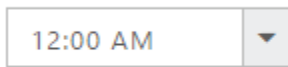
### HTML

```
<input type="text" id="time" />
```

### JAVASCRIPT

```
// Configure TimePicker textbox to restrict editing as follows,  
var timeSample = new ej.TimePicker($("#timepick"), {  
    readOnly: true  
});
```

The following screenshot illustrates a **TimePicker** textbox configured to restrict editing.



### Rounded Corner

You can customize the shape of the **TimePicker** widget from regular rectangular shape to rounded rectangle shape using **showRoundedCorner** property that is set to **false** by default.

#### *Configure Rounded corner to TimePicker Text box*

The following steps explain you to change the edges of the textbox to rounded corner.

In the **HTML** page, add a **<input>** element to configure **TimePicker** widget.

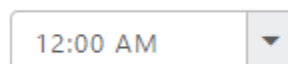
#### **HTML**

```
<input type="text" id="time" />
```

#### **JAVASCRIPT**

```
// You can configure Rounded Corner in TimePicker control as follows,  
var timeSample = new ej.TimePicker($("#timepick"), {  
  showRoundedCorner: true,  
});
```

The following screenshot illustrates a **TimePicker** when **showRoundedCorner** is set to **"true"**.



### Scaling

**TimePicker** widget provides you options to change its height, width and also to change popup height and width.

#### *Change TimePicker widget Height and width*

You can use **height** and **width** property to customize **TimePicker** width and height.

In the **HTML** page, add a **<input>** element to configure **TimePicker** widget.

#### **HTML**

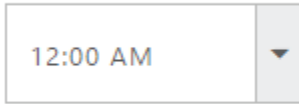
```
<input type="text" id="time" />
```

#### **JAVASCRIPT**

```
// You can customize the width and height of the TimePicker as follows.  
$(function () {  
  var timeSample = new ej.TimePicker($("#timepick"), {  
    width: 150,  
    height: 50  
  });  
});
```

```
});
```

Execute the above code to render the following output.



#### *Changing TimePicker PopupHeight and PopupWidth*

You can use **popupHeight** and **popupWidth** property to customize **TimePicker popup** width and height.

In the **HTML** page, add a **<input>** element to configure **TimePicker** widget.

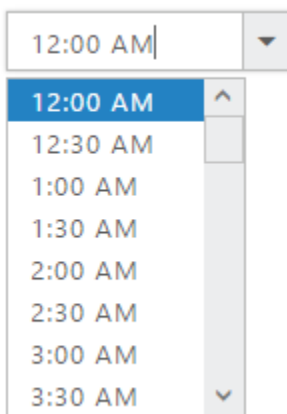
#### **HTML**

```
<input type="text" id="time" />
```

#### **JAVASCRIPT**

```
// Use popupHeight and popupWidth property for the TimePicker as follows.  
$(function () {  
  var timeSample = new ej.TimePicker($("#timepick"), {  
    popupHeight: 170,  
    popupWidth: 120  
  });  
});
```

Execute the above code to render the following output.



#### *State persistence*

**TimePicker** widget provide options to maintain the selected value after you refresh the page by using **enablePersistence** property.

#### *Steps to use State persistence of the TimePicker*

The following steps explains you to use **enablePersistence** property.

In the **HTML** page, add a **<input>** element to configure **TimePicker** widget.

## HTML

```
<input type="text" id="time" />
```

## JAVASCRIPT

```
// Use enablePersistence property to maintain selected value as follows.
$(function () {
  var timeSample = new ej.TimePicker($("#timepick"), {
    value: "10:00 AM",
    enablePersistence: true
  });
});
```

### Strict mode of the TimePicker

**TimePicker** widget provides you an option to set default value when there is no default value between minTime and maxTime by using **enableStrictMode** property.

#### Steps to use StrictMode property

The following steps explains you to use strict mode property.

In the **HTML** page, add a **<input>** element to configure **TimePicker** widget.

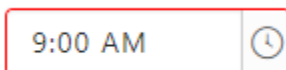
## HTML

```
<input type="text" id="time" />
```

## JAVASCRIPT

```
// Use enableStrictMode property to set default value as follows.
$(function () {
  var timeSample = new ej.TimePicker($("#timepick"), {
    value: "9:00 AM",
    enableStrictMode: true,
    minTime: "10:00 AM",
    maxTime: "9:00 PM"
  });
});
```

Execute the above code to render the following output.



### Interval

**TimePicker** widget provides you an option to change the interval of the hour, minute and seconds.

#### Steps to change the Time interval of the TimePicker control

The following steps explains you to change the Time Interval of the **TimePicker**.

In the **HTML** page, add a **<input>** element to configure **TimePicker** widget.

## HTML

```
<input type="text" id="time" />
```

### JAVASCRIPT

```
// Change Time Interval using hourInterval, minuteInterval and  
secondInterval property.  
$(function () {  
    var timeSample = new ej.TimePicker($("#timepick"), {  
        timeFormat: "h:mm:ss tt",  
        hourInterval: 2,  
        minutesInterval: 30,  
        secondsInterval: 20  
    });  
});  
![] (Behaviour-Settings_images/Behaviour-Settings_img9.png)
```

### Range

**TimePicker** widget provide options to set the Range (minTime & maxTime) for the time.

Steps to change minTime & maxTime of the TimePicker

The following steps explains you to change the Range of the **TimePicker**.

In the **HTML** page, add a **<input>** element to configure **TimePicker** widget.

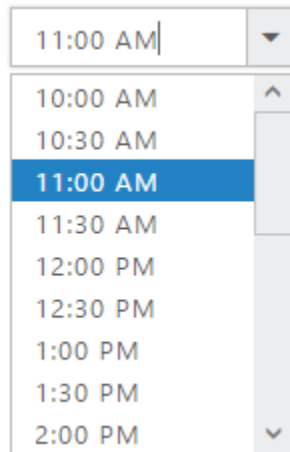
### HTML

```
<input type="text" id="time" />
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module TimePickerComponent {  
    // Use minTime and maxTime property to change the Range of the  
    TimePicker.  
    $(function () {  
        var timeSample = new ej.TimePicker($("#timepick"), {  
            minTime: "10:00 AM",  
            maxTime: "9:00 PM"  
        });  
    });  
}
```

Execute the above code to render the following output.



### TimePicker Customization

The **TimePicker** provides support to display a **TimePicker** in your web page and allows you to pick a time from it.

#### Creating TimePicker Widget

The following steps explain you to create a **TimePicker** widget.

In an **HTML** page, add a **<input>** element to configure **TimePicker** widget.

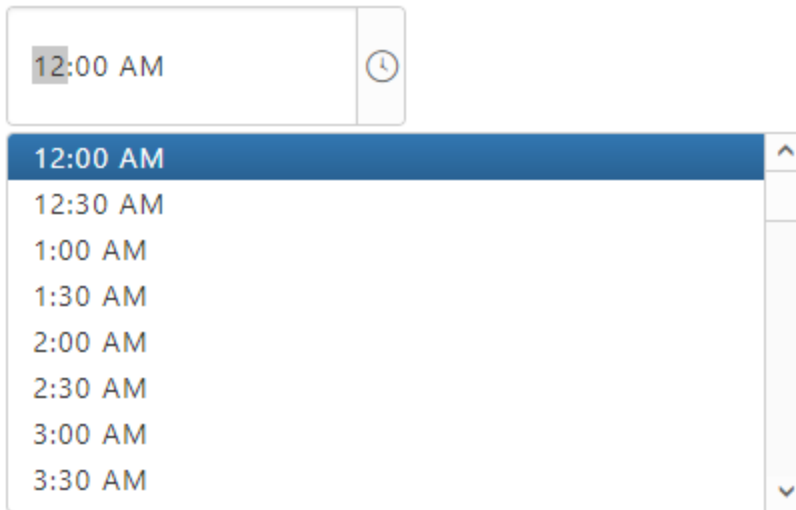
#### HTML

```
<input type="text" id="time" />
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
// You can render TimePicker control as follows.
$(function () {
    var timeSample = new ej.TimePicker($("#timepick"), {
        popupWidth: "400px",
        height: "60px",
        width: "200px"
    });
});
```

The following screenshot illustrates you a default **TimePicker**.



### Time Format

**TimePicker** widget provides you an option to change the time format.

#### Steps to change Time Format of TimePicker widget

The following steps explain you to change the time format for the **TimePicker**.

In the **HTML** page, add a **<input>** element to configure **TimePicker** widget.

#### HTML

```
<input type="text" id="time" />
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TimePickerComponent {
    // Change time format for TimePicker controls as follows.
    $(function () {
        var timeSample = new ej.TimePicker($("#timepick"), {
            timeFormat: "h:mm:ss tt"
        });
    });
}
```

Execute the above code to render the following output.



### Globalization

**TimePicker** supports globalization with different culture.

#### Enabling Globalization Support

The following steps explain you on how to enable **Globalization** support for **TimePicker**.

In an **HTML** page, add a **<input>** element to configure **TimePicker** widget.

**HTML**

```
<input type="text" id="time" />
```

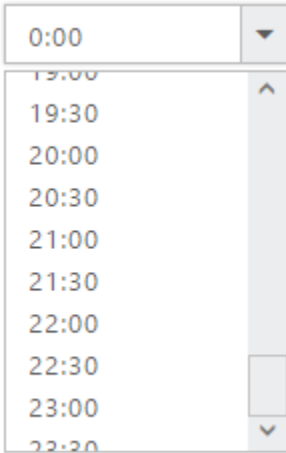
**HTML**

```
<html>
<head>
<!--You can enable localization property using the following code
example.-->
<title>Essential Studio for JavaScript : Timepicker </title>
<meta name="viewport" content="width=device-width, initial-scale=1.0"
charset="utf-8" />
<!-- Style sheet for default theme (flat azure) -->
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />
<!--Scripts-->
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"></script>
<script src="https://cdn.syncfusion.com/js/assets/i18n/ej.culture.zh-
CN.min.js"></script>
</head>
<body>
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<div class="frame">
<div class="control">
<input type="text" id="time" accesskey="j"/>
</div>
</div>
</div>
</div>
</div>
<script type="text/javascript">
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TimePickerComponent {
$(function () {
var timeSample = new ej.TimePicker($("#timepick"), {
locale: "zh-CN"
});
});
}
</script>
</body>
</html>
```

**Note:** jQuery.easing external dependency has been removed from version 14.3.0.49 onwards. Kindly include this jQuery.easing dependency for versions lesser than 14.3.0.49 in order to support animation effects. jQuery.globalize has been removed from version 13.4.0.53 onwards. For version lower than 13.4.0.53, refer jQuery.globalize.min.js along with ej.web.all.min.js to support globalization.



Execute the above code to render the following output.



### RTL

This feature supports to change the left-to-right alignment of the **TimePicker** widget to right-to-left(RTL). The custom template **TimePicker** also supports RTL.

#### Enabling Right-To-Left Support

The following steps explain you in enabling the right-to-left property for the **TimePicker**.

In the **HTML** page, add a **<input>** element to configure **TimePicker** widget.

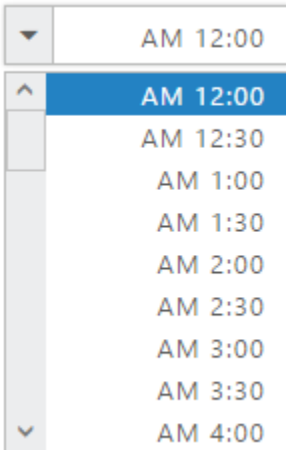
#### HTML

```
<input type="text" id="time" />
```

#### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TimePickerComponent {
  // Enable RTL for TimePicker controls as follows.
  $(function () {
    var timeSample = new ej.TimePicker($("#timepick"), {
      enableRTL: true
    });
  });
}
```

The following screenshot illustrates a **TimePicker** control when **enableRTL** is set to **"true"**



### Keyboard Interaction

You can use **Keyboard** shortcut keys as an alternative to the mouse on using **TimePicker** widget. **TimePicker** widget allows you to perform all kinds of actions using keyboard shortcuts.

List of keyboard shortcuts

| Shortcut Key                   | Description                    |
|--------------------------------|--------------------------------|
| <a href="#">Access key</a> + j | Focuses into TimePicker widget |
| Alt + Down                     | Opens/Hides the popup          |
| Right/Left                     | Moves to adjacent part         |
| Up                             | Increments the value           |
| Down                           | Decrements the value           |

List of keyboard shortcuts, When popup is open

| Shortcut Key | Description                    |
|--------------|--------------------------------|
| Up           | Selects the previous time      |
| Down         | Selects the next time.         |
| Home/End     | Moves to the first / last item |
| Esc          | Closes the popup               |

### Configure Keyboard Interaction

The following steps explain you on how to enable keyboard interaction for the **TimePicker** widget.

In the **HTML** page, add a **<input>** element to configure **TimePicker** widget and enable keyboard interaction by setting the **access key** property.

#### HTML

```
<input type="text" id="time" accesskey="j"/>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TimePickerComponent {
  // You can render the TimePicker control using the following code.
  $(function () {
    var timeSample = new ej.TimePicker($("#timepick"), {
    });
  });
}
```

Run the code sample, press [Access key](#) + J to focus in the TimePicker widget that enables it and you can navigate using arrow keys and Esc key to close the popup.



## ToggleButton

### Overview

The **Toggle Button** allows you to perform the toggle option by using checked and unchecked state. This **Toggle Button** can be helpful to you to check their states. The **Toggle Button** control displays both text and images. The text displayed on the **Toggle Button** is contained in the Text property. The **Toggle Button** control displays images using the sprite CSS Class and **ImagePosition** properties. The **Toggle Button** has theme support.

### Key Features

- **Trendy Look** : Rich appearance with theme Support
- **RTL** : Supports for right to left alignment
- **Text and Image** : Supports both text and image as Toggle Button content in both On/Off state
- **Built-in Icon** : Supports the built-in icon libraries
- **Easy Customization** : The customization of button control to any form is made simple

### Getting Started

Using the following steps, you can create a **TypeScript** ToggleButton component.

#### Creating an ToggleButton in TypeScript

You can create a **TypeScript** application with the help of the given [TypeScript Getting Started Documentation](#).

To create a ToggleButton, include a `input` element with the HTML `id` attribute and pre-defined options set to it.

## HTML

```
<input type="checkbox" id="toggle">
<label for="toggle">Toggle</label>
<script src="app.js"></script>
```

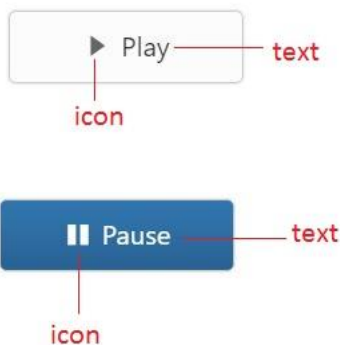
- Create app.ts file and use the below content

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module ToggleButtonComponent {
  var toggleObj = new ej.ToggleButton($("#toggle"), {
    defaultText: "Play",
    activeText: "Pause",
    defaultPrefixIcon: "e-icon e-mediaplay",
    activePrefixIcon: "e-icon e-mediapause",
    showRoundedCorner: true,
    size: "large",
    contentType: "textandimage",
  })
}
```

- Now build your application, so that the **app.ts** file will be compiled and automatically generated the **app.js** file which is added to your project (User have nothing to do with this file). Now, whatever code changes that you make in **app.ts** file will be reflected in app.js file by compiling build the application.

Execution of above code will render the following output.



### Easy Customization

**Toggle Button** is used in all applications. Toggle Button Size, Content and content location is varied according to each application. The following section contains some customizable option for **Toggle Button** that can perform easily.

#### Toggle State

**Toggle Button** has two states like off / on state in a switch. By default you can set any state at initial and then you can move from one state to another state by clicking on the **Toggle Button**. These two states are **Default** and **Active**. **toggleState** property is used to set the state of **Toggle Button** as default state or active state.

The following steps explains you the details about rendering the **Toggle Button** with different toggle state.

In the **HTML** page, add the following button elements to configure **Toggle Button** widget.

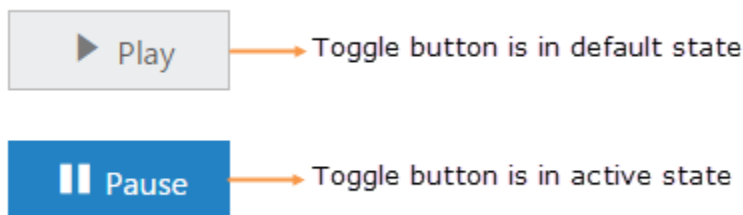
## HTML

```
<input type="checkbox" id="toggle_default" />
<br />
<input type="checkbox" id="toggle_active" />
```

## JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ToggleButtonComponent {
$(function () {
var button = new ej.ToggleButton($("#toggle_default"), {
size: "small",
contentType: "textandimage",
defaultText: "Play",
activeText: "Pause",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause",
//set the button in default state
toggleState: false
});
var button1 = new ej.ToggleButton($("#toggle_active"), {
size: "small",
contentType: "textandimage",
defaultText: "Play",
activeText: "Pause",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause",
//set the button in active state
toggleState: true
});
});
});
}
```

Execute the above code to render the following output.



Toggle state with icons

### Prefix and Suffix Icons

You can add icons in prefix and suffix position of **Toggle Button**. Location of Icon is customized easily using the following mentioned option. This is applicable for the content type's **imageOnly**, **textandimage**, **imagetextimage** and **imageboth**.

**Toggle Button** control also supports the Built-in icon libraries. The **ej.widgets.core.min.css** contains the definition for important icons that are used in toggle buttons. You can use these Built-in icons by

mentioning the icon class name as value in **defaultPrefixIcon**, **defaultSuffixIcon**, **activePrefixIcon**, and **activeSuffixIcon** property. You can use any font icons that is defined in **ej.widgets.core.min.css**. It avoids the complexity in specifying icon using sprite image and CSS.

For example the following mentioned Built-in CSS class are used to show the font icons that are used by media player.

- e-mediaback
- e-mediaforward
- e-medianext
- e-mediaprev
- e-mediaeject
- e-mediaclose
- e-mediapause
- e-mediaplay

#### *Prefix Icon*

It inserts the icon at the starting position of **Toggle Button**. After this prefix icon, you can use text or suffix icon.

#### *Suffix Icon*

It inserts the icon at the ending position of **Toggle Button**. Before this suffix icon, you can use text or prefix icon.

You can also set icon in different location (prefix, suffix) and in different state (default, active) by using the option provided. The following properties are defined for merging the option to add text, icon with different position and in toggle states.

Property Table

| Property Type     | Description   |
|-------------------|---|
| activeText        | Specifies the text of toggle button in active state         |
| activePrefixIcon  | Specifies the prefix icon of toggle button in active state  |
| activeSuffixIcon  | Specifies the suffix icon of toggle button in active state  |
| defaultText       | Specifies the text of toggle button in default state.       |
| defaultPrefixIcon | Specifies the prefix icon of toggle button in default state |
| defaultSuffixIcon | Specifies the suffix icon of toggle button in default state |

The following steps explains you the details about rendering the **Toggle Button** with above mentioned customization properties.

In the **HTML** page, add the following button elements to configure **Toggle Button** widget

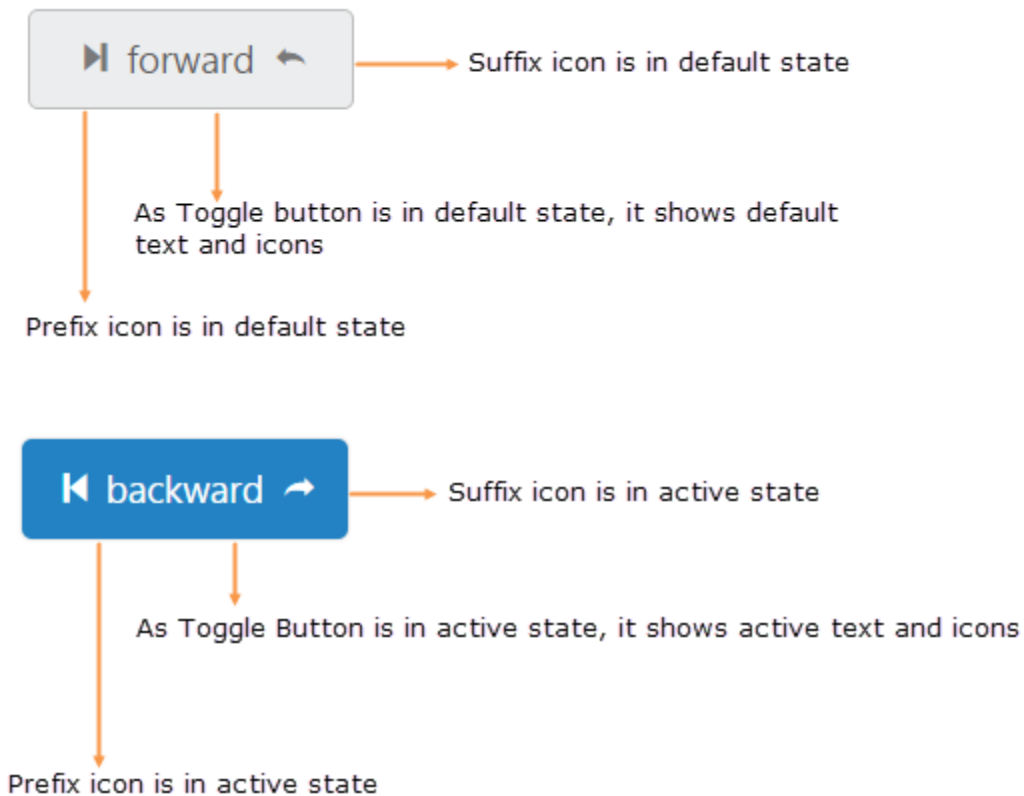
#### **HTML**

```
<input type="checkbox" id="toggle_content" />  
<label for="toggle_content">Toggle</label>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToggleButtonComponent {
$(function () {
var button = new ej.ToggleButton($("#toggle_content"), {
showRoundedCorner: true,
contentType: "imagetextimage",
//e-mediaforward class contain font icon
defaultPrefixIcon: "e-icon e-mediaforward",
//e-mediaback class contain font icon
activePrefixIcon: "e-icon e-mediaback",
defaultText: "forward",
activeText: "backward",
//e-undo class contain font icon
defaultSuffixIcon: "e-icon e-undo",
//e- redo class contain font icon
activeSuffixIcon: "e-icon e-redo"
});
});
}
```

Execute the above code to render the following output.



### Toggle button size

You can render the **Toggle Button** in different sizes. You can use some predefined size option for rendering a **Toggle Button** in easiest way. Each size option has different height and width. It mainly avoids the complexity in rendering **Toggle Button** with complex CSS class. You can mention the **size** of the **Toggle Button** using the following five predefined size options.

#### Predefined Toggle Button size

| Button Size | Description  |
|-------------|--|
| normal      | Creates toggle button with content size.                                 |
| mini        | Creates toggle button with Built-in mini size height, width specified.   |
| small       | Creates toggle button with Built-in small size height, width specified.  |
| medium      | Creates toggle button with Built-in medium size height, width specified. |
| large       | Creates toggle button with Built-in large size height, width specified.  |

You can also set your own width and height for toggle button using height and width property.

The following steps explains you the details about rendering the **Toggle Button** with above mentioned size options.

In the **HTML** page, add the following button elements to configure **Toggle Button** widget.

#### HTML

```
<table>
<tr>
<td class="btnsht">
<input type="checkbox" id="toggle_normal" />
<label for="toggle_normal">Toggle</label>
</td>
</tr>
<tr>
<td class="btnsht">
<input type="checkbox" id="toggle_mini" />
<label for="toggle_mini">Toggle</label>
</td>
</tr>
<tr>
<td class="btnsht">
<input type="checkbox" id="toggle_small" />
<label for="toggle_small">Toggle</label>
</td>
</tr>
<tr>
<td class="btnsht">
<input type="checkbox" id="toggle_medium" />
<label for="toggle_medium">Toggle</label>
</td>
</tr>
<tr>
<td class="btnsht">
```



```

<input type="checkbox" id="toggle_large" />
<label for="toggle_large">Toggle</label>
</td>
</tr>
<tr>
<td class="btnsht">
<input type="checkbox" id="toggle_customSize" />
<label for="toggle_customSize">Toggle</label>
</td>
</tr>
</table>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToggleButtonComponent {
$(function () {
var button = new ej.ToggleButton($("#toggle_normal"), {
//normal size of toggle button
size: "normal",
showRoundedCorner: true,
contentType: "imageonly",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause",
});
var button1 = new ej.ToggleButton($("#toggle_mini"), {
defaultText: "Play",
showRoundedCorner: true,
activeText: "Next",
//mini size of toggle button
size: "mini",
});
var button2 = new ej.ToggleButton($("#toggle_small"), {
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
//small size of toggle button
size: "small",
});
var button3 = new ej.ToggleButton($("#toggle_medium"), {
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
//medium size of toggle button
size: "medium",
});
var button4 = new ej.ToggleButton($("#toggle_large"), {
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
//large size of toggle button
size: "large",
contentType: "textandimage",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause",
});
});
}

```

```

});
var button5 = new ej.ToggleButton($("#toggle_customSize"), {
  showRoundedCorner: true,
  defaultText: "Play",
  activeText: "Next",
  //set own height and width
  height: 50,
  width: 150,
  contentType: "textandimage",
  defaultPrefixIcon: "e-icon e-mediaplay",
  activePrefixIcon: "e-icon e-mediapause",
});
});
}

```

Configure the styles

### CSS

```

<style>
.control {
width: 600px;
}
</style>

```

Execute the above code to render the following output.



### Content type

The content of the **Toggle Button** is mainly text and images. Instead of using complex CSS classes to render **Toggle Button** with different content types, you can use some predefined content type options as listed in the following table. Using this content type you can easily add different types of content for **Toggle Button**. The **Toggle Button** supports the following content types.

List of Content types

| Content Type | Description |
|--------------|-------------|
|--------------|-------------|

|                |  |
|----------------|--|
| textonly       | Supports only for text content only.               |
| imageonly      | Supports only for image content only               |
| imageboth      | Supports image for both ends of the toggle button. |
| textandimage   | Supports image with the text content.              |
| imagetextimage | Supports image with both ends and middle in text.  |

The following steps explain you the details about rendering the **Toggle Button** with above mentioned **content type** options.

In the **HTML** page, add the following button elements to configure **Toggle Button** widget.

### HTML

```
<table>
<tr>
<td class="btnsht">
<input type="checkbox" id="toggle_imageonly" />
<label for="toggle_imageonly">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_textonly" />
<label for="toggle_textonly">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_imageboth" />
<label for="toggle_imageboth">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_textandimage" />
<label for="toggle_textandimage">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_imagetextimage" />
<label for="toggle_imagetextimage">Toggle</label>
</td>
</tr>
</table>
<br/>
<br/>
<table>
<tr>
<td class="btnsht">
<input type="checkbox" id="toggle_imageonly_small" />
<label for="toggle_imageonly_small">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_textonly_small" />
<label for="toggle_textonly_small">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_imageboth_small" />
<label for="toggle_imageboth_small">Toggle</label>
</td>
</tr>
</table>
```

```

</td>
<td class="btnsht">
<input type="checkbox" id="toggle_textandimage_small" />
<label for="toggle_textandimage_small">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_imagetextimage_small" />
<label for="toggle_imagetextimage_small">Toggle</label>
</td>
</tr>
</table>

```

## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToggleButtonComponent {
$(function () {
var button = new ej.ToggleButton($("#toggle_imageonly"), {
showRoundedCorner: true,
//only image is used as content
contentType: "imageonly",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button1 = new ej.ToggleButton($("#toggle_textonly"), {
//only text is used as content
contentType: "textonly",
defaultText: "Play",
showRoundedCorner: true,
activeText: "Next"
});
var button2 = new ej.ToggleButton($("#toggle_imageboth"), {
//only images in both end is used as content
contentType: "imageboth",
showRoundedCorner: true,
defaultPrefixIcon: "e-icon e-mediaforward",
activePrefixIcon: "e-icon e-mediaback",
defaultSuffixIcon: "e-icon e-undo",
activeSuffixIcon: "e-icon e-redo"
});
var button3 = new ej.ToggleButton($("#toggle_textandimage"), {
//text and image is used as content
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button4 = new ej.ToggleButton($("#toggle_imagetextimage"), {
//image text image is used as content
contentType: "imagetextimage",
showRoundedCorner: true,
defaultPrefixIcon: "e-icon e-mediaforward",
activePrefixIcon: "e-icon e-mediaback",

```

```

defaultText: "forward",
activeText: "backward",
defaultSuffixIcon: "e-icon e-undo",
activeSuffixIcon: "e-icon e-redo"
});
var button5 = new ej.ToggleButton($("#toggle_imageonly_small"), {
size: "small",
showRoundedCorner: true,
contentType: "imageonly",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button6 = new ej.ToggleButton($("#toggle_textonly_small"), {
size: "small",
contentType: "textonly",
defaultText: "Play",
showRoundedCorner: true,
activeText: "Next"
});
var button7 = new ej.ToggleButton($("#toggle_imageboth_small"), {
size: "small",
contentType: "imageboth",
showRoundedCorner: true,
defaultPrefixIcon: "e-icon e-mediaforward",
activePrefixIcon: "e-icon e-mediaback",
defaultSuffixIcon: "e-icon e-undo",
activeSuffixIcon: "e-icon e-redo"
});
var button8 = new ej.ToggleButton($("#toggle_textandimage_small"), {
size: "small",
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button9 = new ej.ToggleButton($("#toggle_imagetextimage_small"), {
size: "small",
contentType: "imagetextimage",
showRoundedCorner: true,
defaultPrefixIcon: "e-icon e-mediaforward",
activePrefixIcon: "e-icon e-mediaback",
defaultText: "forward",
activeText: "backward",
defaultSuffixIcon: "e-icon e-undo",
activeSuffixIcon: "e-icon e-redo"
});
});
}

```

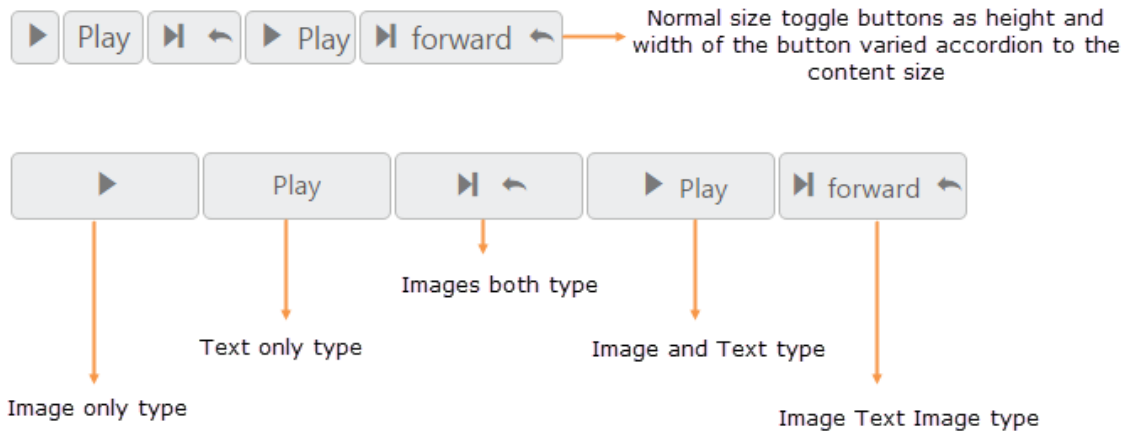
Configure the styles

### CSS

```
<style>
```

```
.normal {
width: 300px;
}
.small {
width: 450px;
}
</style>
```

Execute the above code to render the following output.



### Image position

To provide the best look and feel for **Toggle Button**, position of images in toggle button is important. You can easily customize the position of images inside toggle button using **imagePosition** property without using any complex CSS. **imagePosition** property is applicable only with the **textandimage** content type property. This property represent the position of images with respect to text.

### Property Table

| Image Position | Description   |
|----------------|---|
| imageleft      | Support for aligning text in right and image in left. |
| imageright     | Support for aligning text in left and image in right. |
| imagetop       | Support for aligning text in bottom and image in top. |
| imagebottom    | Support for aligning text in top and image in bottom. |

The following steps explains you the details about rendering the **Toggle Button** with above mentioned image Position options.

In the **HTML** page, add the following button elements to configure **Toggle Button** widget.

### HTML

```
<table>
<tr>
<td class="btnsht">
```

```

<input type="checkbox" id="toggle_imageleft_normal" />
<label for="toggle_imageleft_normal">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_imageleft_mini" />
<label for="toggle_imageleft_mini">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_imageleft_small" />
<label for="toggle_imageleft_small">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_imageleft_medium" />
<label for="toggle_imageleft_medium">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_imageleft_large" />
<label for="toggle_imageleft_large">Toggle</label>
</td>
</tr>
</table>
<br />
<br />
<table>
<tr>
<td class="btnsht">
<input type="checkbox" id="toggle_imageright_normal" />
<label for="toggle_imageright_normal">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_imageright_mini" />
<label for="toggle_imageright_mini">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_imageright_small" />
<label for="toggle_imageright_small">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_imageright_medium" />
<label for="toggle_imageright_medium">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_imageright_large" />
<label for="toggle_imageright_large">Toggle</label>
</td>
</tr>
</table>
<br />
<br />
<table>
<tr>
<td class="btnsht">
<input type="checkbox" id="toggle_imagetop" />
<label for="toggle_imagetop">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_imagebottom" />

```

```
<label for="toggle_imagebottom">Toggle</label>
</td>
</tr>
</table>
```

## JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToggleButtonComponent {
$(function () {
var button = new ej.ToggleButton($("#toggle_imageleft_normal"), {
size: "normal",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button1 = new ej.ToggleButton($("#toggle_imageleft_mini"), {
size: "mini",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button2 = new ej.ToggleButton($("#toggle_imageleft_small"), {
size: "small",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button3 = new ej.ToggleButton($("#toggle_imageleft_medium"), {
size: "medium",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button4 = new ej.ToggleButton($("#toggle_imageleft_large"), {
size: "large",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
```



```
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button5 = new ej.ToggleButton($("#toggle_imageright_normal"), {
size: "normal",
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button6 = new ej.ToggleButton($("#toggle_imageright_mini"), {
size: "mini",
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button7 = new ej.ToggleButton($("#toggle_imageright_small"), {
size: "small",
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button8 = new ej.ToggleButton($("#toggle_imageright_medium"), {
size: "medium",
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button8 = new ej.ToggleButton($("#toggle_imageright_large"), {
size: "large",
imagePosition: "imageright",
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button9 = new ej.ToggleButton($("#toggle_imagetop"), {
imagePosition: "imagetop",
```

```
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause",
width: 60
});
var button10 = new ej.ToggleButton($("#toggle_imagebottom"), {
imagePosition: "imagebottom",
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause",
width: 60
});
});
}
```

Configure the styles

### CSS

```
<style>
.one {
width: 450px;
}
.two{
width: 115px;
}
</style>
```

Execute the above code to render the following output.



### Theme support

You can control the style and appearance of **Toggle Button** based on CSS classes. To apply styles to the **Toggle Button** control, you can refer two files, **`ej.widgets.core.min.css`** and **`ej.theme.min.css`**. When you refer **`ej.widgets.all.min.css`** file, then it is not necessary to include the files **`ej.widgets.core.min.css`** and **`ej.theme.min.css`** in your project, as **`ej.widgets.all.min.css`** is the combination of these two.

By default, there are 16 theme support available for RadialMenu component, namely:

- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap
- high-contrast-01
- high-contrast-02
- material
- office-365

### Custom CSS

You can use the CSS to customize the **Toggle Button** control appearance. Define a CSS class as per requirement and assign the class name to **`cssClass`** property.

The following steps explain you the details about rendering the **Toggle Button** with custom CSS.

In the **HTML** page, add the following button elements to configure **Toggle Button** widget.

### HTML

```
<table>
<tr>
<td class="btnsht">
<input type="checkbox" id="toggle_customCss1" />
<label for="toggle_customCss1">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_customCss2" />
<label for="toggle_customCss2">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_customCss3" />
<label for="toggle_customCss3">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_customCss4" />
<label for="toggle_customCss4">Toggle</label>
</td>
<td class="btnsht">
<input type="checkbox" id="toggle_customCss5" />
<label for="toggle_customCss5">Toggle</label>
</td>
</tr>
</table>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToggleButtonComponent {
$(function () {
var button = new ej.ToggleButton($("#toggle_customCss1"), {
cssClass: "customCss1",
size: "small",
showRoundedCorner: true,
contentType: "textandimage",
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
var button1 = new ej.ToggleButton($("#toggle_customCss2"), {
cssClass: "customCss2",
size: "small",
showRoundedCorner: true,
contentType: "textandimage",
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
});
```

```

var button2 = new ej.ToggleButton($("#toggle_customCss3"), {
  cssClass: "customCss3",
  size: "small",
  showRoundedCorner: true,
  contentType: "textandimage",
  defaultText: "Play",
  activeText: "Next",
  defaultPrefixIcon: "e-icon e-mediaplay",
  activePrefixIcon: "e-icon e-mediapause"
});
var button3 = new ej.ToggleButton($("#toggle_customCss4"), {
  cssClass: "customCss4",
  size: "small",
  showRoundedCorner: true,
  contentType: "textandimage",
  defaultText: "Play",
  activeText: "Next",
  defaultPrefixIcon: "e-icon e-mediaplay",
  activePrefixIcon: "e-icon e-mediapause"
});
var button4 = new ej.ToggleButton($("#toggle_customCss5"), {
  cssClass: "customCss5",
  size: "small",
  showRoundedCorner: true,
  contentType: "textandimage",
  defaultText: "Play",
  activeText: "Next",
  defaultPrefixIcon: "e-icon e-mediaplay",
  activePrefixIcon: "e-icon e-mediapause"
});
});
}

```

Configure the CSS styles to apply on buttons.

### CSS

```

<style type="text/css" class="cssStyles">
/* Customize the button background */
.e-togglebutton.customCss1 {
background-color: #121111;
}
.e-togglebutton.customCss2 {
background-color: #94bbd5;
}
.e-togglebutton.customCss3 {
background-color: #f3533c;
}
.e-togglebutton.customCss4 {
background-color: #d1eed;
}
.e-togglebutton.customCss5 {
background-color: #deb66e;
}
/* Customize the button image & text color */

```

```
.e-togglebutton.customCss1.e-btn.e-select .e-icon, .e-
togglebutton.customCss1.e-btn.e-select .e-btntxt {
color: #94bbd5;
}
.e-togglebutton.customCss2.e-btn.e-select .e-icon, .e-
togglebutton.customCss2.e-btn.e-select .e-btntxt {
color: #121111;
}
.e-togglebutton.customCss3.e-btn.e-select .e-icon, .e-
togglebutton.customCss3.e-btn.e-select .e-btntxt {
color: #cef6f7;
}
.e-togglebutton.customCss5.e-btn.e-select .e-icon, .e-
togglebutton.customCss5.e-btn.e-select .e-btntxt {
color: #534f4f;
}
</style>
```

Execute the above code to render the following output.



## Button types

**Toggle Button** is used as normal click able button, submitting form data, resetting the form data to its initial value. According to the usage of button, you can render the **Toggle Button** in the following three types by using the **type** property.

### Toggle Button Types

| Button Type | Description   |
|-------------|---|
| button      | The button is a click able button   |
| submit      | The button is a submit button (submits form-data)                         |
| reset       | The button is a reset button (resets the form-data to its initial values) |

The following steps explains you the details about rendering the **Toggle Button** with above mentioned types.

In the **HTML** page, add the following button elements to configure **Toggle Button** widget.

### HTML

```
<table>
<tr>
<td class="btnsht">
<input type="checkbox" id="type_button" />
</td>
</tr>
<tr>
<td class="btnsht">
<input type="checkbox" id="type_submit" />
```

```

</td>
</tr>
<tr>
<td class="btnsht">
<input type="checkbox" id="type_mini" />
</td>
</tr>
</table>

```

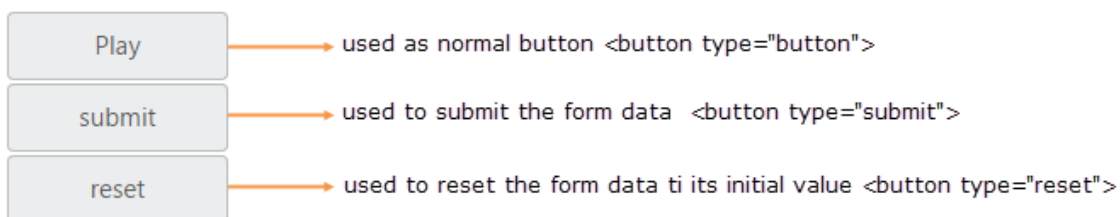
## JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToggleButtonComponent {
$(function () {
var button = new ej.ToggleButton($("#type_button"), {
size: "large",
type: "button",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
});
var button1 = new ej.ToggleButton($("#type_submit"), {
size: "large",
type: "submit",
showRoundedCorner: true,
defaultText: "submit",
activeText: "submitted",
});
var button2 = new ej.ToggleButton($("#type_mini"), {
size: "large",
type: "reset",
showRoundedCorner: true,
defaultText: "reset",
activeText: "reseted",
});
});
}

```

Execute the above code to render the following output.



## RTL support

In some cases, it is necessary to use right to left alignment. You can render **RTL** support by using **enableRTL** property. In **RTL** mode, when there is more than one content (image/text, image/image) in

button, then the content is aligned in right to left format. For example, when text is in left and image is in right position, after applying right to left alignment these positions are interchanged.

The following steps explain you the details about rendering the **Toggle Button** with Right to left alignment support.

In the **HTML** page, add the following button elements to configure **Toggle Button** widget.

#### HTML

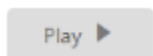
```
<input type="checkbox" id="toggle_rtl" />
```

#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToggleButtonComponent {
$(function () {
var button = new ej.ToggleButton($("#toggle_rtl"), {
size: "small",
imagePosition: "imageleft",
contentType: "textandimage",
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: " e-icon e-mediaplay",
activePrefixIcon: " e-icon e-mediapause",
//used to enable or disable RTL support for toggle button
enableRTL: true
});
});
}
```

In above mentioned code example **prefixIcon** property is used and the icon that is to be on left side, (before text) is rendered on right side as **enableRTL** property is used with **prefixIcon**. Consequently, the alignment is changed in right to left order.

Output of above steps



## Miscellaneous

### Show Rounded Corner

It sets the corner of **Toggle Button** in rounded shape. The **Toggle Button**, by default doesn't have rounded corner. To set rounded corner, you can enable **showRoundedCorner** property.

The following steps explain you the details about rendering the **Toggle Button** with Rounded corner support.

In the **HTML** page, add the following button elements to configure **Toggle Button** widget.

#### HTML

```
<input type="checkbox" id="toggle_roundedCorner" />
```



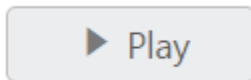
**HTML**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ToggleButtonComponent {
$(function () {
var button = new ej.ToggleButton($("#toggle_roundedCorner"), {
width: "100px",
height: "30px",
contentType: "textandimage",
//used to specify the rounded corner for toggle button
showRoundedCorner: true,
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: "e-icon e-mediaplay",
activePrefixIcon: "e-icon e-mediapause"
});
});
}

```

Execute the above code to render the following output.

**Prevent Toggle**

This property is used to prevent the state change of **Toggle Button** when it is clicked. When you set **preventToggle** property as true, then the state of the **Toggle Button** is not changed even though it is clicked.

The following steps explains you the details about rendering the **Toggle Button** with **preventToggle** property enabled.

In the **HTML** page, add the following button elements to configure **Toggle Button** widget.

**HTML**

```
<input type="checkbox" id="toggle_prevent" />
```

**HTML**

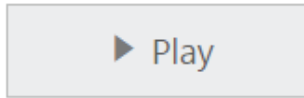
```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module ToggleButtonComponent {
$(function () {
var button = new ej.ToggleButton($("#toggle_roundedCorner"), {
size: "large",
contentType: "textandimage",
defaultText: "Play",
activeText: "Next",
defaultPrefixIcon: " e-icon e-mediaplay",
activePrefixIcon: " e-icon e-mediapause",
//prevent changing state of toggle button
preventToggle: true
});
});
}

```

```
} } ;  
}
```

Execute the above code to render the following output.



The state of the Toggle button is not changed, When you click on it. As prevent toggle property is set as true

## Toolbar

### Overview

The Toolbar control supports displaying a list of tools within a web page. This control is capable of customizing toolbar items with any functionality by using enriched client-side methods. This control is composed of collection of unordered lists containing text and images contained into a div.

### Key Features

- **Modern look:** Rich appearance with theme support.
- **RTL:** Supports right-to-left alignment.
- **Text and Image:** Supports both text and images as toolbar content.
- **Easy Customization:** The customization of toolbar control to any form is made simple.
- **Data binding:** Supports for data binding with local data and remote data.
- **Template:** Supports for Customizing toolbar with user needed tools.
- **Keyboard navigation:** Support for navigation through keyboard.

### Getting Started

This section explains briefly about how to create a **Toolbar** in your application with **Typescript**.

#### Create Toolbar for PDF Reader

**Toolbar** control supports displaying a list of tools in a Web page. This control is capable of customizing toolbar items with any functionality by using enriched **client-side** methods. This control consists of a collection of **unordered lists** contains text and images into a **<div>**. From the following section, you can learn how to customize **toolbar** control for a **PDF reader** scenario. The following screen shot shows the appearance of **toolbar** in **PDF reader** simulator application.

![[/js/Toolbar/Getting-Startedimages/Getting-Startedimg1.png)

#### Create a Toolbar

The **Essential TypeScript Toolbar** control can be easily configured with **HTML <DIV>** and **<UL><LI>** elements. The basic rendering of **Essential JS Toolbar** is achieved by the default functionality.

Create an HTML file and add the following template into the **HTML** file for **Toolbar** creation.

#### HTML

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="utf-8" />  
<title>Getting Started - RichTextEditor</title>
```

```
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"></script>
</head>
</html>
```

Add div or span element for **Toolbar** rendering.

### HTML

```
<div id="ToolbarItem"> </div>
```

Initialize the Toolbar in ts file by using the `ej.Toolbar` method.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToolbarComponent {
$(function () {
var sample = new ej.Toolbar($("#ToolbarItem"), {});
});
}
```

Output of the above steps



### Initialize Toolbar Items

**Toolbar** consists of a list of items. From the following section, you can learn how to initialize the toolbar items with **UL LI** template.

Initialize the Toolbar items with **UL LI** template as follows.

### HTML

```
<div class="control">
<div id="ToolbarItem">
<!--list of toolbar items-->
<ul>
<li id="OtherFormat" title="Convert PDF files to Word or Excel Online..">
<div class="PdfDocument e-icon convertToOthers "></div>
</li>
<li id="PDFOnline" title="Convert files to PDF Online">
<div class="PdfDocument e-icon convertToPdf "></div>
</li>
<li id="Signature" title="Sign, add text or send a document for
signature">
<div class=" PdfDocument e-icon signature "></div>
</li>
<li id="Save" title="Save file ( Ctrl+S )">
<div class=" PdfDocument e-icon save "></div>
</li>
```

```
<li id="Print" title="Print file ( Ctrl+P ) ">
<div class=" PdfDocument e-icon print "></div>
</li>
<li id="Message" title="Message">
<div class=" PdfDocument e-icon msg "></div>
</li>
</ul>
</div>
</div>
```

Apply the given styles in the code table to show the **toolbar items** as follows. You can refer images from any location. In the following code sample, the images are referred from the given location.

<http://js.syncfusion.com/UG/Web/Content/>

### CSS

```
<style type="text/css" class="cssStyles">
.e-tooltxt .PdfDocument.e-icon {
background-image: url('http://js.syncfusion.com/UG/Web/Content/pdf-
icon.png');
background-repeat: no-repeat;
display: block;
height: 30px;
width: 30px;
}
.e-tooltxt .PdfDocument.e-icon: hover {
background-image: url('http://js.syncfusion.com/UG/Web/Content/pdf-icon-
white.png');
}
.PdfDocument.e-icon.convertToOthers {
background-position: -349px 0px;
}
.PdfDocument.e-icon.convertToPdf {
background-position: -527px 0px;
}
.PdfDocument.e-icon.signature {
background-position: 2px 0px;
}
.PdfDocument.e-icon.save {
background-position: -87px 0px;
}
.PdfDocument.e-icon.msg {
background-position: -483px 0px;
}
</style>
```

After updating the **Toolbar items** with their **CSS** styles, you can render the toolbar inside **<script>** tag.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToolbarComponent {
$(function () {
var sample = new ej.Toolbar($("#ToolbarItem"), {
width: "auto",
```

```
height: "33px",
});
});
}
```

Execute the code to render a toolbar with a list of **toolbar items**.

![[/js/Toolbar/Getting-Startedimages/Getting-Startedimg3.png)

Toolbar with list of toolbar items

Render remaining Toolbar items

In the above output only few **toolbar items** are rendered, but you need to render all the **toolbar items** to achieve the requirements. You can separate or group the toolbar items. The separation or grouping of toolbar items is achieved when you give toolbar items as a list of **UL LI** values inside the toolbar **<div>** or **span** element. From the following section, you can learn how to initialize the remaining toolbar items with **UL LI** template and how to group the toolbar items.

Initialize the Toolbar items with **UL LI** template as follows.

### HTML

```
<div id="ToolbarItem">
<!--Initializes toolbar items from above code example -->
<!-- Separator is added at the end of each ul inside the toolbar element-
->
<!-- list of Remaining toolbar items with item separator -->
<ul>
<li id="Previous" title="Show previous page ( Left Arrow )">
<div class=" PdfDocument e-icon previous "></div>
</li>
<li id="Next" title="Show next page ( Right Arrow )">
<div class="PdfDocument e-icon next "></div>
</li>
<li id="page">
<div class="PdfDocument">
<input type="text" value="1" />
</div>
</li>
<li id="count">
<span>/ 1</span>
</li>
</ul>
<ul>
<li id="ZoomOut" title="Zoom Out">
<div class=" PdfDocument e-icon zoomOut "></div>
</li>
<li id="ZoomIn" title="Zoom In">
<div class=" PdfDocument e-icon zoomIn "></div>
</li>
<li id="ZoomValue">
<div class=" PdfDocument">
<!-- input element for rendering Zoom value dropdown -->
<input type="text" id="selectPercent" />
</div>
</li>
```

```

</ul>
<ul>
<li id="FitFull" title="Fit one full page to window">
<div class=" PdfDocument e-icon fitOne "></div>
</li>
<li id="StickyNote" title="Add stick note ( Ctrl+6 ) ">
<div class=" PdfDocument e-icon sticky "></div>
</li>
<li id="ReadMode" title="View File in Read Mode">
<div class=" PdfDocument e-icon readMode "></div>
</li>
</ul>
</div>

```

Add the following styles in the code table to display the toolbar items as follows.

### CSS

```

<style>
.PdfDocument.e-icon.previous {
background-position: -395px 0px;
}
.PdfDocument.e-icon.next {
background-position: -439px 0px;
}
.PdfDocument.e-icon.zoomIn {
background-position: -175px 0px;
}
.PdfDocument.e-icon.zoomOut {
background-position: -219px 0px;
}
.PdfDocument.e-icon.fitOne {
background-position: -264px 0px;
}
.PdfDocument.e-icon.sticky {
background-position: -131px -1px;
}
.PdfDocument.e-icon.readMode {
background-position: -308px 0px;
}
.PdfDocument.e-icon.print {
background-position: -43px 0px;
}
#ZoomValue .PdfDocument {
width: 90px;
}
#page .PdfDocument input {
text-align: center;
width: 20px;
height: 21px;
}
#count span {
width: 30px;
height: 30px;
position: relative;
top: 2px;
}

```

```
text-align: center;
vertical-align: middle;
}
</style>
```

After updating the Toolbar items with their **CSS** styles, you can render the toolbar inside the **<script>** tag and also need to render the drop down list control for **select zoom value**. Basically, dropdown list control is rendered with input element. **Set Zoom value** is one of the items in the toolbar. The following code example shows how to render and initialize **drop down control** with list of **zoom values**.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToolbarComponent {
$(function () {
var percentList = ["10%", "25%", "50%", "100%", "400%", "800%", "1600%",
"3200%", "6400%"];
var toolbar = new ej.Toolbar($("#ToolbarItem"), {
width: "auto",
height: "33px",
});
var dropdownlist = new ej.DropDownList($("#selectPercent"), {
width: "90px",
height: "27px",
dataSource: percentList,
value: "100%"
});
});
});
}
```

Execute the code to render a **toolbar items** with separator.



### Add Actions to Toolbar Items

Now the **toolbar** is rendered so you need to render the header and content area to create a **PDF reader**. From the following section, you can learn how to render the **header** (Toolbar), **contentsection** (PDF viewer area) and how to set the action to toolbar items.

You are not going to deal with PDF reading or rendering task here. You will only simulate the PDF Reader app to demonstrate the Toolbar control usage and will completely ignore the PDF rendering area.

Initialize the content area and header as specified in the code table.

### HTML

```
<!-- control class used for aligns the pdf reader in center of a page. -->
<div class="control">
<div class="ctrllabel"></div>
<!-- Here Initialize the Toolbar items as like above code sample -->
<div id="contentSection">
<textarea id="content" rows="10" cols="30">
Description:
```

Toolbar control supports displaying a list of tools within a Web page. This control is capable of customizing toolbar items with any functionality by using enriched client-side methods. This control is composed of collection of unordered lists containing text and images contained into a div.

```
</textarea>
</div>
</div>
```

You can apply the following styles with the above styles to design the **PDF header** and **content area**. The desired output is shown as follows.

### CSS

```
<style type="text/css" class="cssStyles">
#content {
float: left;
height: 300px;
width: 628px;
position: absolute;
}
.control {
margin: 110px 320px 0;
position: relative;
}
.ctrllabel {
background-image: url("http://js.syncfusion.com/UG/Web/Content/pdf-
header.png");
background-repeat: no-repeat;
width: 634px;
height: 32px;
}
</style>
```

Execute the given code to render a **PDF reader** as follows.

```

```

So far, you have added the required toolbar items and configured its appearance. When you click on **toolbar items**, the operation is performed through **client side click event**. The following code example explains how to perform operations when you click on the **toolbar items**.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToolbarComponent {
$(function () {
var sample = new ej.Toolbar($("#ToolbarItem"), {
width: "auto",
height: "33px",
enableSeparator: true,
click: "onItemClick"
});
});
});
```



```

function onItemClick(args) {
//Finds Out the Item that was Clicked in Toolbar
//args.currentTarget returns the clicked Toolbar element
var option = args.currentTarget.id; //Finds Out the Id of Clicked item.
switch (option) {
case "OtherFormat":
//writes a code for Convert pdf files to Other format.
case "PDFOnline":
//writes a code for Convert files to Pdf online.
case "Signature":
//writes a code for Send a document for signature.
case "Save":
//writes a code for Save content.
case "Print":
//writes a code for Print content.
case "Message":
//writes a code for Send a Message.
case "Previous":
//writes a code for Show previous page.
case "Next":
//writes a code for Show Next page.
case "ZoomOut":
//writes a code for Zoom out the page.
case "ZoomIn":
//writes a code for Zoom In the page.
case "FitFull":
//writes a code for Fit one full page to window.
case "StickyNote":
//writes a code for Add Sticky Note.
case "ReadMode":
//writes a code for view file in read mode.
}
}
}

```

## Data binding

**Toolbar** provides you a flexible approach for binding data from various data sources. There are various properties in **Toolbar** for Data Binding.

### Data fields and configuration

The following sub-properties provides a way to bind either the local/remote data to the **Toolbar** control.

Property Table of JavaScript Toolbar control

| Property   | Syntax                                | Description   |
|------------|---------------------------------------|---|
| dataSource | dataSource: window.countriesField     | It specifies the data source of the Toolbar. The data source contains the list of data for generating the Toolbar items. By default, its value is null and its data type is object. |
| field      | fields: {text: "name", value: "key" } | It specifies the mapping fields for the data items of the Toolbar. By   |

|                 |   |   |
|-----------------|---|---|
|                 |   | default, its value is null and its data type is object.   |
| query           | query:<br>ej.Query().from("Suppliers").select("ContactName"); | It specifies the query to retrieve the data from online server. By default, its value is null and its data type is object.  |
| id              | id  | It specifies the id of the tag and its default value is null and data type is string.   |
| text            | text  | It specifies the text content of the tag and its default value is null and data type is string.   |
| imageUrl        | imageUrl  | This property defines the imageUrl for the image location. While setting images, the folder name in which the images are stored should be set to imageUrl property. The value set to this property should be string type. |
| imageAttributes | imageAttributes   | This property defines style for the image. While setting an image, styles can be applied such as height, width by using this property. The value set to this property should be string type.                              |
| spriteCssClass  | spriteCssClass  | This property sets the Sprite CSS for the image tag in Toolbar. The value set to this property should be string type.   |
| htmlAttributes  | htmlAttributes  | This property sets the HTML attribute for the Toolbar item. The value set to this property should be object type. It can be any HTML attribute such as id, class, style.  |
| tooltipText     | tooltipText   | This property sets the text value for Toolbar item while mouse over in Toolbar. The value set to this property should be string type.   |

### Local data

**Toolbar** provides you an extensive data binding support to generate **Toolbar** items, that the values can be mapped to the **ToolBar** fields, namely **key** and **text**.

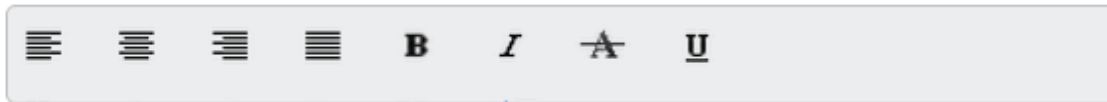
And also you can add image, image styles, sprite CSS class, query and HTML attributes options with data binding fields. The following code explains the details about the data binding with **Toolbar**.

### HTML

```
<div class="cols-sample-area">
<div id="toolbar"></div>
</div>
```

### CSS

```
<style type="text/css" class="cssStyles">
.darktheme .cols-sample-area .e-tooltxt .ToolbarItems {
background-image: url('../images/toolbar/ui-icons-metro.png');
}
.cols-sample-area .e-tooltxt .ToolbarItems {
display: block;
background-image: url('../images/toolbar/ui-icons-dark.png');
height: 22px;
width: 22px;
}
.e-tooltxt:hover .ToolbarItems, .darktheme .cols-sample-area .e-
tooltxt:hover .ToolbarItems {
background-image: url('../images/toolbar/ui-icons-light.png');
}
.ToolbarItems.LeftAlign_tool {
background-position: -26px -39px;
}
.ToolbarItems.CenterAlign_tool {
background-position: -55px -39px;
}
.ToolbarItems.RightAlign_tool {
background-position: -89px -39px;
}
.ToolbarItems.Justify_tool {
background-position: -123px -39px;
}
.ToolbarItems.Bold_tool {
background-position: -159px -39px;
}
.ToolbarItems.Italic_tool {
background-position: -196px -39px;
}
.ToolbarItems.StrikeThrough_tool {
background-position: -55px -70px;
}
.ToolbarItems.Underline_tool {
background-position: -23px -68px;
}
.html {
background-color: yellowgreen;
}
</style>
```



### Remote data

You can bind the data for the **Toolbar** items from remote. That is, you can access the data from any other server that is located as remote web service. It uses DataManager to retrieve data and you can create DataManager with its **URL**. To create DataManager, define ej.DataManager() to a variable. Then provide online link to **url** property. Assign this variable to **Toolbar** dataSource.

To bind Remote data to **Toolbar**, use the following code example.

### HTML

```
<div id="toolbar"></div>
```

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToolbarComponent {
    $(function () {
        // DataManager creation
        var dataManger = ej.DataManager({
            url: "http://mvc.syncfusion.com/Services/Northwnd.svc/"
        });
        // Query creation
        var query = ej.Query()
            .from("Orders").take(6);
        var sample = new ej.Toolbar($("#toolbar"), {
            dataSource: dataManger,
            fields: { text: "CustomerID" },
            query: query,
            orientation: "horizontal",
            width: "340px"
        });
    });
}
```

VINET TOMSP HANAR VICTE SUPRD HANAR

### Behavior settings

The following are some miscellaneous properties that enables you to change the behavior of **Toolbar** control.

#### Enabling Toolbar

The **Toolbar** property **enabled** is used to enable or disable the **Toolbar**. Set the value to this property as **Boolean** type.

**HTML**

```

<div class="cols-sample-area">
<div id="toolbar">
<ul>
<li id="Left" title="Left">
<div class="ToolbarItems LeftAlign_tool"></div>
</li>
<li id="Center" title="Center">
<div class="ToolbarItems CenterAlign_tool"></div>
</li>
<li id="Right" title="Right">
<div class="ToolbarItems RightAlign_tool"></div>
</li>
<li id="Justify" title="Justify">
<div class="ToolbarItems Justify_tool"></div>
</li>
</ul>
<ul>
<li id="Bold" title="Bold">
<div class="ToolbarItems Bold_tool"></div>
</li>
<li id="Italic" title="Italic">
<div class="ToolbarItems Italic_tool"></div>
</li>
<li id="StrikeThrough" title="Strike Through">
<div class="ToolbarItems StrikeThrough_tool"></div>
</li>
<li id="UnderLine" title="UnderLine">
<div class="ToolbarItems Underline_tool"></div>
</li>
</ul>
</div>
</div>

```

**JAVASCRIPT**

```

$(function () {
// declaration
var sample = new ej.Toolbar($("#toolbar"), {
enabled: true
});
});

```

**CSS**

```

<style type="text/css" class="cssStyles">
.darktheme .cols-sample-area .e-tooltxt .ToolbarItems {
background-image: url('../images/toolbar/ui-icons-metro.png');
}
.cols-sample-area .e-tooltxt .ToolbarItems {
display: block;
background-image: url('../images/toolbar/ui-icons-dark.png');
height: 22px;
width: 22px;
}

```

```
.e-tooltxt:hover .ToolbarItems, .darktheme .cols-sample-area .e-
tooltxt:hover .ToolbarItems {
background-image: url('../images/toolbar/ui-icons-light.png');
}
.ToolbarItems.LeftAlign_tool {
background-position: -26px -39px;
}
.ToolbarItems.CenterAlign_tool {
background-position: -55px -39px;
}
.ToolbarItems.RightAlign_tool {
background-position: -89px -39px;
}
.ToolbarItems.Justify_tool {
background-position: -123px -39px;
}
.ToolbarItems.Bold_tool {
background-position: -159px -39px;
}
.ToolbarItems.Italic_tool {
background-position: -196px -39px;
}
.ToolbarItems.StrikeThrough_tool {
background-position: -55px -70px;
}
.ToolbarItems.Underline_tool {
background-position: -23px -68px;
}
}
</style>
```

### Hiding Toolbar

The **Toolbar** property **hide** is used to show or hide the **Toolbar**. Set the value to this property as **Boolean** type.

#### JAVASCRIPT

```
$(function () {
// declaration
var sample = new ej.Toolbar($("#toolbar"), {
hide: true
});
});
```

### Orientation

The **Toolbar** control supports both vertical and horizontal orientations, allowing it to fit into any scenario. The **orientation** property of **Toolbar** defines the orientation in which the control is rendered. Set the value to this property as **enum or string** type. It accepts the following values.

- ej.Orientation.Horizontal or "Horizontal"
- ej.Orientation.Vertical or "Vertical"

The following section explains you on how to set orientation for the toolbar.

### Horizontal

This property sets the **Toolbar** in **horizontal** orientation. You can refer the following steps to set horizontal orientation for **Toolbar** control.

In View, create UL-LI list of toolbar items and invoke the toolbar helper.

Add the following script in your **HTML** page.

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToolbarComponent {
$(function () {
var sample = new ej.Toolbar($("#toolbar"), {
width: "290px",
orientation: ej.Orientation.Horizontal
});
});
}
```

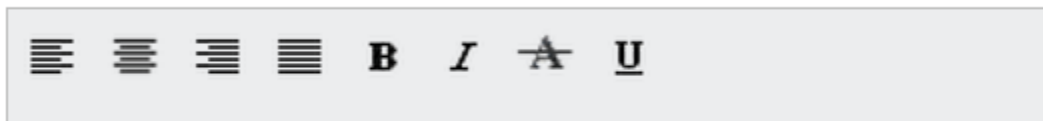
OR

#### JS

```
$(function () {
// declaration
var sample = new ej.Toolbar($("#toolbar"), {
width: "290px",
orientation: "Horizontal"
});
});
```

Build and run the application.

The following screenshot illustrates a **Toolbar** with horizontal orientation.



### Vertical

This property sets the **Toolbar** in **vertical** orientation. You can refer the following steps to set vertical orientation for **Toolbar** control.

Create UL-LI list of toolbar items and invoke the toolbar helper.

Add the following script in your **HTML** page.

#### JS

```
$(function () {
// declaration
var sample = new ej.Toolbar($("#toolbar"), {
```

```
orientation: ej.Orientation.Vertical  
});  
});
```

OR

JS

```
$(function () {  
  // declaration  
  var sample = new ej.Toolbar($("#toolbar"), {  
    orientation: "Vertical"  
  });  
});
```

Build and run the application.

The following screenshot illustrates a **Toolbar** with vertical orientation.





## Appearance and Styling

### Adjusting Toolbar size

#### Height

The **height** property is used to set height of the **Toolbar**. Set the value to this property as **number** or **string** type.

#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToolbarComponent {
$(function () {
var sample = new ej.Toolbar($("#toolbar"), {
height: 300
});
});
});
}
```

#### Width

The **width** property is used to set width of the **Toolbar**. Set the value to this property as **number** or **string** type.

#### HTML

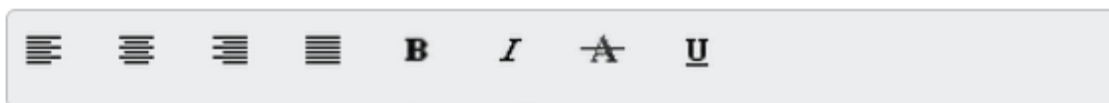
```
$(function () {
// declaration
var sample = new ej.Toolbar($("#toolbar"), {
height: "300px"
});
});
```

### Enabling Rounded Corner

The **showRoundedCorner** property is used to enable rounded corner for **Toolbar**. Set the value to this property as **Boolean** type.

#### JAVASCRIPT

```
$(function () {
// declaration
var sample = new ej.Toolbar($("#toolbar"), {
showRoundedCorner: true
});
});
```

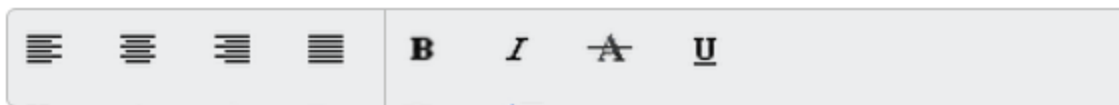


### Enabling Separator

The **enableSeparator** property is used to set separator between **Toolbar** items. It separates one or more list items. When you want to separate two set of items, then define them in two separate 'ul' tags and set the value to this property as **Boolean** type.

#### JAVASCRIPT

```
$(function () {  
  // declaration  
  var sample = new ej.Toolbar($("#toolbar"), {  
    enableSeparator: true  
  });  
});
```



### Themes

You can control the **style and appearance of the Toolbar** based on **CSS** classes. In order to apply styles to the Toolbar control, you can refer two files - **ej.widgets.core.min.css** and **ej.theme.min.css**. When you refer **ej.widgets.all.min.css** file, it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.widgets.all.min.css** is the combination of these two.

By default, there are 16 theme support available for RadialMenu component, namely:

- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap
- high-contrast-01
- high-contrast-02
- material
- office-365

### CssClass

The **cssClass** property is used to set root class for **Toolbar** control theme. Set the value to this property as **string** type.

#### JAVASCRIPT

```
$(function () {
  // declaration
  var sample = new ej.Toolbar($("#toolbar"), {
    width: "290px",
    cssClass: "gradient-lime"
  });
});
```

### CSS

```
<style>
.gradient-lime {
background-color: yellowgreen;
}
</style>
```



### Template Support

Template allows you to insert custom controls inside the toolbar items. Also you can design simple drop down buttons listing the items and radio button inside the **Toolbar**.

Set the list for **DropDown control** inside a list tag and define this tag as a **Toolbar** item. You can use all simple controls as a **Toolbar** item. To add RadioButton and DropDownList to **Toolbar**, use the following code example.

### HTML

```
<div id="toolbar">
<ul>
<li>
<div>
<input type="radio" name="small" id="Radio1" />
</div>
Option 1
</li>
<li id="Dropdown" title="Dropdown Control">
<input id="selectcar" type="text" />
</li>
</ul>
</div>
```

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToolbarComponent {
  $(function () {
    // declaration
```

```

var skills = [
{ skill: "ASP.NET" }, { skill: "ActionScript" }, { skill: "Basic" },
{ skill: "C++" }, { skill: "C#" }, { skill: "dBase" }, { skill: "Delphi"
},
{ skill: "ESPOL" }, { skill: "F#" }, { skill: "FoxPro" }, { skill: "Java"
},
{ skill: "J#" }, { skill: "Lisp" }, { skill: "Logo" }, { skill: "PHP" }
];
var sample1 = new ej.RadioButon($("#Radio1"),{
size:"medium"
});
var sample2 = new ej.DropDownList($("#selectcar"),{
dataSource: skills,
fields: { text: "skill" },
watermarkText: "Select your skill"
});
var sample3 = new ej.ejToolbar($("#toolbar"),{
width: "250px",
height: "28px"
});
});
}

```

Through Items API:

#### HTML

```
<div id="toolbar"></div>
```

#### JS

```

$(function () {
var skills = [
{ skill: "ASP.NET" }, { skill: "ActionScript" }, { skill: "Basic" },
{ skill: "C++" }, { skill: "C#" }, { skill: "dBase" }, { skill: "Delphi"
},
{ skill: "ESPOL" }, { skill: "F#" }, { skill: "FoxPro" }, { skill: "Java"
},
{ skill: "J#" }, { skill: "Lisp" }, { skill: "Logo" }, { skill: "PHP" }
];
var ToolbarInstance = new ej.Toolbar($("#toolbar"),{
Items:[
{id:"item1",group:"group1",template:"<input type='radio' id='Radio1'
name='radio'>Option 1</input>"},
{id: "item2",group:"group2",template:"<input type='text' id='dropdown1'
/>" }
],
});
var RadioInstance = new ej.DropDownList($("#Radio1"),{
size:"medium"
});
var DropDownListInstance = new ej.DropDownList($("#dropdown1"),{
dataSource: skills,
fields: { text: "skill" },
watermarkText: "Select your skill",
});

```

```
});
```

Through template field in dataSource API:

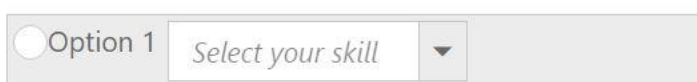
### HTML

```
<div id="toolbar"></div>
```

### JS

```
$(function () {
    var skills = [
        { skill: "ASP.NET" }, { skill: "ActionScript" }, { skill: "Basic" },
        { skill: "C++" }, { skill: "C#" }, { skill: "dBase" }, { skill: "Delphi" },
        { skill: "ESPOL" }, { skill: "F#" }, { skill: "FoxPro" }, { skill: "Java" },
        { skill: "J#" }, { skill: "Lisp" }, { skill: "Logo" }, { skill: "PHP" }
    ];
    toolbaritems = [
        {
            id: "1",
            title: "radio",
            template: "<input type='radio' id='radio1'>Option 1</input>",
            groups: "group1"
        },
        {
            id: "2",
            title: "dropdown",
            template: "<input type='text' id='dropdown2'></div>",
            group: "group2"
        }
    ],
    var sample = new ej.Toolbar($("#toolbar"), {
        dataSource: toolbaritems,
        fields: { id:
            "id", tooltipText: "title", group: "group", template: "template" },
    });
    var sample1 = new ej.RadioButton($("#radio1"), {
        size: "medium"
    });
    var sample2 = new ej.DropDownList($("#dropdown2"), {
        dataSource: skills,
        fields: { text: "skill" },
        watermarkText: "Select your skill",
    });
});
```

The following screenshot displays a Toolbar with embedded controls.



## RTL

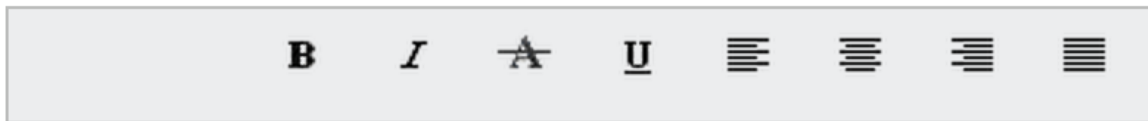
This feature allows you to change the left-to-right alignment of the **Toolbar** to right-to-left (**RTL**) that sets the **Toolbar** to do its actions from right to left. The **enableRTL** property sets the **Toolbar** from right to left. Set the value to this property as **Boolean** type.

## JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToolbarComponent {
$(function () {
var sample = new ej.Toolbar($("#toolbar"), {
width: "290px",
enableRTL: true
});
});
}

```



## Keyboard Navigation

The entire **Toolbar** commands can be accessed through the keyboard by specifying the **Keyboard Shortcut** listed in the following table.

List of keyboard shortcuts

| Keyboard Shortcut | Function                        |
|-------------------|---------------------------------|
| Alt + j           | Focuses the control             |
| UP                | Navigates up and left.          |
| Down              | Navigates down and right.       |
| Left              | Navigates up and left.          |
| Right             | Navigates down and right.       |
| Home              | Navigates to the starting item. |
| End               | Navigates to the ending item.   |
| Enter             | Selects the focused item        |

The following code example illustrates shortcuts associated with the **Toolbar** items.

## HTML

```

<!-- Refer Local Data section for style and data bound for toolbar item -
-->

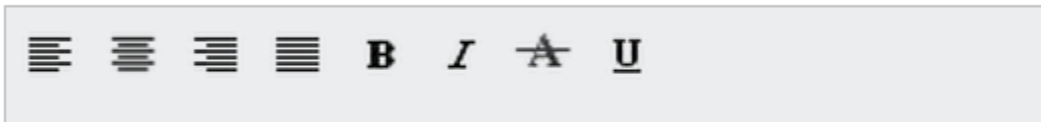
```

**JS**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToolbarComponent {
$(function () {
  // declaration
  var sample = new ej.Toolbar($("#toolbar"), {
    width: "290px"
  });
  //Control focus key
  $(document).on("keydown", function (e) {
    if (e.altKey && e.keyCode === 74) { // j- key code.
      $("#toolbar").focus();
    }
  });
});
}

```

**Responsive Layout**

**Responsive Layout** is aimed at crafting sites to provide an optimal viewing experience easy reading and navigation with a minimum of resizing, panning, and scrolling—across a wide range of devices (from mobile phones to desktop computer monitors). You can achieve **Responsive Layout** by using default functionality of **Toolbar** with **isResponsive** as **true** and also you need to give **Toolbar** width as in percentage value and add **ej.responsive.css** file in this sample. **CDN** link for the responsive CSS file is as follows.

[<http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/responsive-css/ej.responsive.css>](<http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/responsive-css/ej.responsive.css>)

Add the above **css** link in the code sample.

Add the following code example in your **HTML** page.

**HTML**

```

<div class="control">
  <!--list of toolbar items-->
  <div id="ToolbarItem">
    <ul>
      <li id="OtherFormat" title="Convert PDF files to Word or Excel Online..">
        <div class="PdfDocument e-icon convertToOthers "></div>
      </li>
      <li id="PDFOnline" title="Convert files to PDF Online">
        <div class="PdfDocument e-icon convertToPdf "></div>
      </li>
    </ul>
  </div>

```

```

<li id="Signature" title="Sign, add text or send a document for
signature">
<div class=" PdfDocument e-icon signature "></div>
</li>
<li id="Save" title="Save file ( Ctrl+S )">
<div class=" PdfDocument e-icon save "></div>
</li>
<li id="Print" title="Print file ( Ctrl+P ) ">
<div class=" PdfDocument e-icon print "></div>
</li>
<li id="Message" title="Message">
<div class=" PdfDocument e-icon msg "></div>
</li>
</ul>
<ul>
<li id="Previous" title="Show previous page ( Left Arrow )">
<div class=" PdfDocument e-icon previous "></div>
</li>
<li id="Next" title="Show next page ( Right Arrow )">
<div class="PdfDocument e-icon next "></div>
</li>
<li id="page">
<div class="PdfDocument">
<input type="text" value="1" />
</div>
</li>
<li id="count">
<span>/ 1</span>
</li>
</ul>
<ul>
<li id="ZoomOut" title="Zoom Out">
<div class=" PdfDocument e-icon zoomOut "></div>
</li>
<li id="ZoomIn" title="Zoom In">
<div class=" PdfDocument e-icon zoomIn "></div>
</li>
<li id="ZoomValue">
<div class=" PdfDocument">
<input type="text" id="selectPercent" />
</div>
</li>
</ul>
<ul>
<li id="FitFull" title="Fit one full page to window">
<div class=" PdfDocument e-icon fitOne "></div>
</li>
<li id="StickyNote" title="Add stick note ( Ctrl+6 ) ">
<div class=" PdfDocument e-icon sticky "></div>
</li>
<li id="ReadMode" title="View File in Read Mode">
<div class=" PdfDocument e-icon readMode "></div>
</li>
</ul>
</div>
</div>

```



**CSS**

```
<style type="text/css" class="cssStyles">
.e-tooltxt .PdfDocument.e-icon {
background-image: url('http://js.syncfusion.com/UG/Web/Content/pdf-
icon.png');
background-repeat: no-repeat;
display: block;
height: 30px;
width: 30px;
}
.e-tooltxt .PdfDocument.e-icon:hover {
background-image: url('http://js.syncfusion.com/UG/Web/Content/pdf-icon-
white.png');
}
.PdfDocument.e-icon.convertToOthers {
background-position: -349px 0px;
}
.PdfDocument.e-icon.convertToPdf {
background-position: -527px 0px;
}
.PdfDocument.e-icon.signature {
background-position: 2px 0px;
}
.PdfDocument.e-icon.save {
background-position: -87px 0px;
}
.PdfDocument.e-icon.msg {
background-position: -483px 0px;
}
.PdfDocument.e-icon.previous {
background-position: -395px 0px;
}
.PdfDocument.e-icon.next {
background-position: -439px 0px;
}
.PdfDocument.e-icon.zoomIn {
background-position: -175px 0px;
}
.PdfDocument.e-icon.zoomOut {
background-position: -219px 0px;
}
.PdfDocument.e-icon.fitOne {
background-position: -264px 0px;
}
.PdfDocument.e-icon.sticky {
background-position: -131px -1px;
}
.PdfDocument.e-icon.readMode {
background-position: -308px 0px;
}
.PdfDocument.e-icon.print {
background-position: -43px 0px;
}
#ZoomValue .PdfDocument {
```

```
width: 90px;
}
#page .PdfDocument input {
text-align: center;
width: 20px;
height: 21px;
}
#count span {
width: 30px;
height: 30px;
position: relative;
top: 2px;
text-align: center;
vertical-align: middle;
}
</style>
```

**JS**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module ToolbarComponent {
$(function () {
var sample = new ej.Toolbar($("#ToolbarItem"), {
width: "600px", // width of the Toolbar
height: "33px", // height of the Toolbar
isResponsive: true // responsive support for toolbar
});
var percentList = ["10%", "25%", "50%", "100%", "400%", "800%", "1600%",
"3200%", "6400%"];
var sample1 = new ej.DropDownList($("#selectPercent"), {
width: "90px",
height: "27px",
dataSource: percentList, // Assign List of Dropdown value to data Source
value: "100%" // Initialize drop down value.
});
});
}
```

Execute the above code to render the following output.



responsiveType:Inline

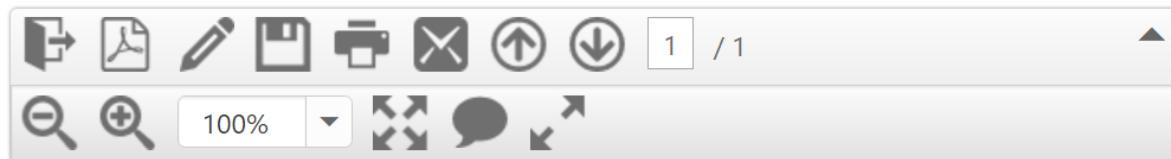
We can set the responsiveType property to inline to display the overflown toolbar items as inline toolbar.

**JS**

```
$(function () {
var sample = new ej.Toolbar($("#ToolbarItem"), {
width: "600px", // width of the Toolbar
```

```
isResponsive: true // responsive support for toolbar
responsiveType:"inline"//responsive Type API to display overflow items as
inline toolbar.
});
});
```

While setting inline responsiveType the following output will be displayed.



[responsiveType:Popup](#)

We can set the responsiveType property to popup to display the overflown toolbar items as popup.

**JS**

```
$(function () {
var sample = new ej.Toolbar($("#ToolbarItem"), {
width: "600px", // width of the Toolbar
height: "33px", // height of the Toolbar
isResponsive: true // responsive support for toolbar
responsiveType:"popup"//responsive Type API to display overflow items as
popup.
});
});
```

While setting popup as responsiveType the following output will be displayed.



## Tooltip

### Overview

The Tooltip control displays a pop-up hint when users hover, click, or focus on an element. The Tooltip will be positioned based on browser window, mouse, target element, or x and y coordinates. The Tooltip's layout can be customized to create a distinct visualized element.

### Key Features

**Responsive** – Flexible positioning of Tooltip and can even reposition themselves when the viewport is resized. **Template** - The Tooltip's layout can be customized to create a distinct visualized element.

**Animation** – Custom animation effects can be used for showing/hiding the Tooltip **RTL support** -

Provides a full-fledged right-to-left mode which aligns content in the Tooltip control from right to left. \*

**Accessibility** - Supports keyboard and ARIA accessibility.

## Getting started

### Preparing HTML document

The Tooltip control has the following list of external JavaScript dependencies.

- [jQuery](#) 1.7.1 and later versions
- [jQuery.easing](#) - to support animation effects in the components

Refer to the internal dependencies in the following table.

| File              | Description/Usage                                       |
|-------------------|---|
| ej.core.min.js    | It is referred always before using all the JS controls. |
| ej.tooltip.min.js | The Tooltip's main file.                                |

To get started, you can use the `ej.web.all.min.js` file that encapsulates all the `ej` controls and frameworks in one single file.

### Create a Tooltip

Create an **HTML** page and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

The Tooltip can be created from any HTML element with the HTML `id` attribute and pre-defined options set to it. To create the Tooltip, you should call the `ejTooltip` jQuery plug-in function with the options as parameter. Refer to the following code example.

#### HTML

```
<div class="frame">
<div class="img" id="link1" >

<div class="desc">Delphi Succinctly</div>
</div>
</div>
```

## HTML

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module TooltipComponent {
$(function () {
var sample1 = new ej.Tooltip($("#link1"), {
content: "Learn the fundamentals of Delphi to build a variety of
solutions for many devices and platforms." });
});
}
```

Apply the following style sheet

## HTML

```
<style>
div.img {
border: 1px solid #ccc;
width: 159px;
height: 213px;
left: 35%;
position: relative;
top: 20%;
}
div.img img {
width: 159px;
height: 179px;
}
div.desc {
padding: 8px;
text-align: center;
}
</style>
```

Learn the fundamentals of Delphi to build a variety of solutions  
for many devices and platforms.



## Setting Dimensions

Tooltip dimensions can be set using [width](#) and [height](#) API.

### HTML

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module TooltipComponent {
$(function () {
var sample1 = new ej.Tooltip($("#link1"), {
content: "Learn the fundamentals of Delphi to build a variety of
solutions for many devices and platforms.",
height: "100px",
width: "100px"
});
});
}
```

## Tooltip Appearance

You can configure the appearance of the Tooltip with the title, close button and call out as your application requires.

### HTML

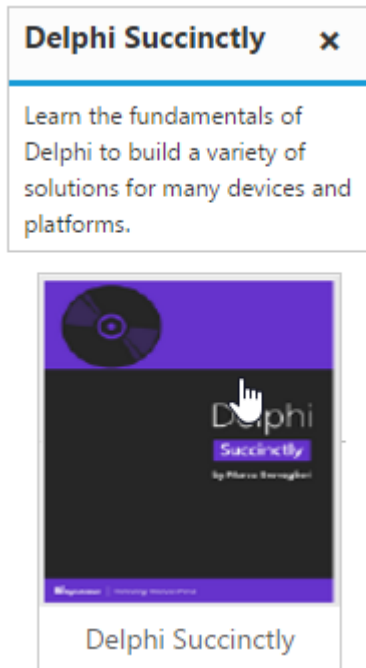
```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module TooltipComponent {
$(function () {
var sample1 = new ej.Tooltip($("#link1"), {
content: "Learn the fundamentals of Delphi to build a variety of
solutions for many devices and platforms.",
height: "100px",
width: "200px",
closeMode: "sticky",
title: "Delphi Succintly",
isBalloon : false
});
});
}
```

Apply the following styles to show the Tooltip.

### HTML

```
<style>
div.img {
border: 1px solid #ccc;
float: left;
box-sizing: border-box;
height: 200px;
width: 146px;
}
div.img img{
width: 100%;
height: 166px;
}
```

```
div.desc {
padding: 6px;
text-align: center;
}
</style>
```



layout: post

title: Customization in Tooltip widget

description: Customization in Tooltip widget

platform: Typescript

control: Tooltip

documentation: ug

keywords : ejTooltip, Tooltip, Tooltip widget, Tooltip template

## Customization

### Template Support

By default you can add any text or image to the Tooltip. To customize the tooltip layout or to create your own visualized elements you can use this template support.

### HTML

```
<div class="ctrl" id="centerImg">

<div class="new">Roslyn Succinctly</div>
</div>
<script type="text/javascript">
```

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TooltipComponent {
$(function () {
var sample = new ej.Tooltip($("#centerImg"), {
{
width: "350px",
content: '<div class="main"> <div class="poster">  </div> <div class="def"> <h4> Roslyn
Succinctly </h4><div class="description">Microsoft has only recently
embraced the world of open source software, offering <a
href="#">More...</a> </div>'
});
});
}
</script>
<style>
h4 {
margin-top: 0px;
margin-bottom: 2px;
}
.e-tooltip-wrap .e-tipContainer .e-tipcontent {
padding: 5px 0px;
}
.poster {
float: left;
padding: 4px 0px;
}
.new {
text-align: center;
}
.def {
float: right;
}
.ctrl {
border: 1px solid #ebebe0;
width: 150px;
padding: 5px;
height: 180px;
margin-top: 239px;
margin-left: 250px;
}
.ctrImg {
width: 150px;
height: 160px;
}
.category {
margin-left: 10px;
}
.description {
width: 200px;
height: 60px;
line-height: 22px;
margin-top: 10px;
}
</style>

```





#### *Tooltip's title customization*

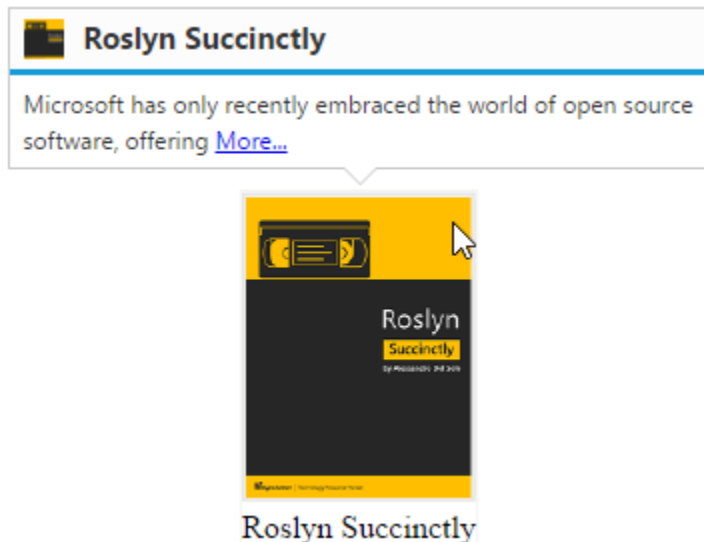
Tooltip title can be customized with the image or any HTML element.

#### **HTML**

```
<div class="ctrl" id="centerImg">

<div class="new">Roslyn Succinctly</div>
</div>
<script type="text/javascript">
var sample = new ej.Tooltip($("#centerImg"), {
{
title : '<div>
<div class="description"> Roslyn Succinctly </div> </div> ',
content: '<div>Microsoft has only recently embraced the world of open
source software, offering <a href="#">More...</a> </div>'
}});
</script>
<style>
.titleImg {
width: 20px;
height: 20px;
float: left;
margin-right: 10px;
}
#centerImg{
margin-left : 300px;
```

```
margin-top : 250px;
position : absolute;
border: 1px solid grey;
}
.description {
height: 20px;
}
</style>
```



### Animation Effects

Determines the type of effect that takes place when showing/hiding the tooltip.

We can specify the effect and the duration for the animation.

| Effects        | Description  |
|----------------|--|
| Slide          | Sliding animation effect takes place with a duration of 200ms. |
| Fade           | Fading animation effect takes place with a duration of 800ms.  |
| None (Default) | No effect takes place  |

Let's create a Tooltip that slides down when shown using the [animation](#) property:

### HTML

```
<div class="control">
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
</div>
// Creates the Tooltip
<script>
var sample = new ej.Tooltip($("#test"), {
content: "JavaScript is the programming language of HTML and the Web.",
animation : { effect : "slide", speed : 1000 }
});
```

```
</script>
```

### Custom Animation

Custom animation effect for both Tooltip show/hide can also be done by [show](#) and [hide](#) method.

Show or Hide method may receive an optional 'callback' parameter, which represents a function you'd like to call which will animate the tooltip.

### HTML

```
<div class="control">
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
</div>
<button id="open">Open</button>
// Creates the Tooltip
<script>
var sample = new ej.Tooltip($("#test"), {
content: "JavaScript is the programming language of HTML and the Web."
});
var sample1 = new ej.Button($("#open"), {
size: "large",
showRoundedCorner: true,
click: "onClick"
});
function onClick(args) {
tip = $("#test").data("ejTooltip");
tip.show(null, "myFunc");
}
function myFunc(args) {
tip = $("#test").data("ejTooltip");
$(tip.tooltip).slideDown(200, "easeOutElastic");
}
</script>
```

**Note:** Show or Hide method can also receive an optional parameter "effect name", (e.g any easing effect name) which specifies the type of effect taken place when showing/hiding of the tooltip, please refer to the following link for online demo - [link](#).

### Modernize the tooltip's content

It's easy to update a tooltip's content – whether it's open or closed.

### HTML

```
<div class="control">
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
<button id="open">Update Content</button>
</div>
<script type="text/javascript">
$(function () {
var sample = new ej.Tooltip($("#test"), {
content: "JavaScript is the programming language of HTML and the Web."
});
var sample1 = new ej.Button($("#open"), {
size: "large",
```

```

showRoundedCorner: true,
click: "onClick"
});
});
function onClick(args){
tip = $("#test").data("ejTooltip");
tip.setModel({ content: "JavaScript" });
tip.show();
}
</script>

```

### Closing Mode

By default, the Tooltip will be hidden when mouse leaves the target element. Different types of close mode as follows

| Types          | Description  |
|----------------|--|
| Auto           | Tooltip will be hidden after a particular period of time.    |
| Sticky         | Tooltip rendered with the button, it will close the tooltip. |
| None (Default) | Tooltip will be hidden when mouse leaves the target element. |

### Auto

The tooltip will be visible only for the period of time specified in the [autoCloseTimeout](#).

Let see an example, this Tooltip will only hide after hovering the target for 2000ms

### HTML

```

<div class="control">
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
</div>
// Creates the Tooltip
<script>
var sample = new ej.Tooltip($("#test"),{
content: "JavaScript is the programming language of HTML and the Web.",
closeMode : "auto",
autoCloseTimeout : 2000
});
</script>

```

---

**Note:** Time specified in the autoCloseTimeout will be in milliseconds and the default value is 4000ms

---

### Sticky

A close button will be shown with the Tooltip. The button element (i.e. close button) located by default at the top right of the Tooltip or title bar (if title is enabled). The tooltip gets closed when the button is clicked.

### HTML

```

<div class="control">

```

```
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
```

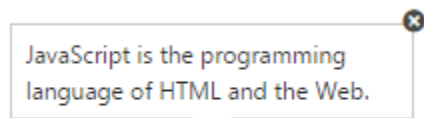
```
</div>
```

```
// Creates the Tooltip
```

```
<script>
```

```
var sample = new ej.Tooltip($("#test"), {
  content: "JavaScript is the programming language of HTML and the Web.",
  width : "200px",
  closeMode : "sticky"
});
```

```
</script>
```



TypeScript lets you write [JavaScript](#) the way you really want to.

You can also have Tooltip with a title, in which case the button will lye within it:

### HTML

```
<div class="control">
```

```
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
```

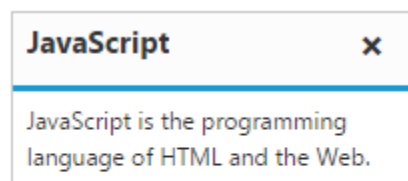
```
</div>
```

```
// Creates the Tooltip
```

```
<script>
```

```
var sample = new ej.Tooltip($("#test"), {
  content: "JavaScript is the programming language of HTML and the Web.",
  width : "200px",
  title : "JavaScript",
  closeMode : "sticky"
});
```

```
</script>
```



TypeScript lets you write [JavaScript](#) the way you really want to.

### Position

The position object defines various attributes of the Tooltip position, including the element it is positioned in relation to, and how the position is adjusted within the defined container.

Lets position the Tooltips (stems) left center corner at the right center of the target element.

### HTML

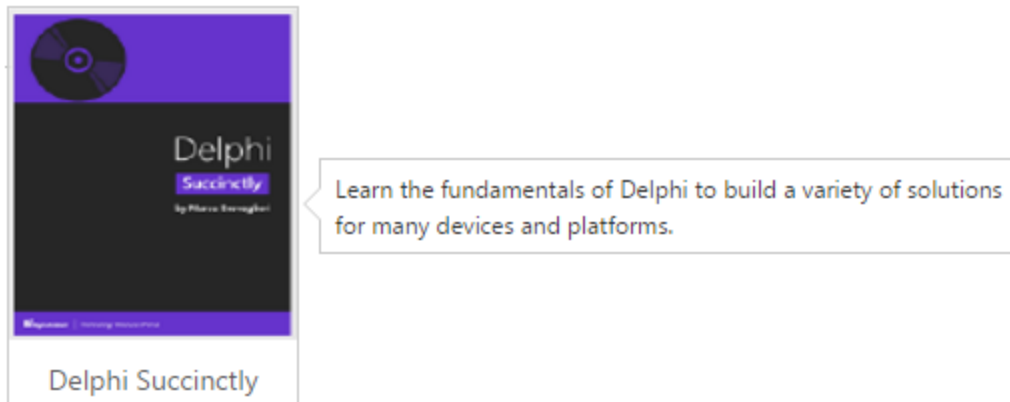
```
<div class="img" id="sample">
<a target="_blank" href="image/taj.png">

</a>
<div class="desc">Delphi Succinctly</div>
</div>
// Creates the Tooltip
<script>
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TooltipComponent {
$(function () {
var sample1 = new ej.Tooltip($("#sample"),{
content: "Learn the fundamentals of Delphi to build a variety of
solutions for many devices and platforms.",
position : {
stem : {horizontal : "left", vertical : "center"},
target : {horizontal: "right", vertical: "center"}
}
});
});
}
</script>
```

Apply the following styles to show the Tooltip.

#### HTML

```
<style>
div.img {
border: 1px solid #ccc;
float: left;
box-sizing: border-box;
height: 200px;
width: 146px;
}
div.img img{
width: 100%;
height: 166px;
}
div.desc {
padding: 6px;
text-align: center;
}
</style>
```



**Note:** By default, the Tooltips "center bottom" corner is placed at the center top of the target element.

### Containment

Determines the HTML element in which the Tooltip is appended to e.g. its containing element and The tooltip will be restricted to move only within the specified container element.

Let's append our Tooltip to a custom 'frame' container:

### HTML

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
</div>
</div>
// Creates the Tooltip
<script>
var sample1 = new ej.Tooltip($("#test"), {
content: "JavaScript is the programming language of HTML and the Web.",
containment: ".frame"
});
</script>
```

**Note:** By default all Tooltips are appended to the document.body element.

### Associates

The Tooltip will be positioned in relation to target element. Can also be set to 'mouse' or the window, or an absolute x/y position on the page.

Let's position the Tooltip in relation to the 'a' element inside the div element:

### HTML

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
</div>
</div>
// Creates the Tooltip
<script>
```

```
var sample1 = new ej.Tooltip($("#test"), {
  content: "JavaScript is the programming language of HTML and the Web."
});
</script>
```

We can also position the Tooltip in relation to the mouse.

### HTML

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
</div>
</div>
// Creates the Tooltip
<script>
var sample1 = new ej.Tooltip($("#test"), {
  content: "JavaScript is the programming language of HTML and the Web.",
  associate : "mousefollow"
});
</script>
```

Position the tooltip at the current mouse position, once enter to the target element as follows

### HTML

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
</div>
</div>
// Creates the Tooltip
<script>
var sample1 = new ej.Tooltip($("#test"), {
  content: "JavaScript is the programming language of HTML and the Web.",
  associate : "mouseenter"
});
</script>
```

It also possible to place the tooltip relation to the window as follows

### HTML

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
</div>
</div>
// Creates the Tooltip
<script>
var sample1 = new ej.Tooltip($("#test"), {
  content: "JavaScript is the programming language of HTML and the Web.",
  associate : "window",
});
</script>
```



```
position : {
target : {horizontal : "right", vertical: "bottom"}
}
});
</script>
```

And last but not least, absolute positioning via X,Y co-ordinates e.g. a Tooltip at 10px from left and top of the page:

### HTML

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
</div>
</div>
// Creates the Tooltip
<script>
var sample1 = new ej.Tooltip($("#test"),{
content: "JavaScript is the programming language of HTML and the Web.",
associate : "axis",
position : {
target : {horizontal : 10, vertical: 10}
}
});
</script>
```

### Collision

When the positioned element overflows the window in some direction, move it to an alternative position.

The following values determines the kind of positioning that takes place.

| Value            | Description   |
|------------------|---|
| Flip             | Flips the element to the opposite side of the target if the collision detected. |
| Fit              | Shift the element away from the edge of the window.                             |
| FlipFit(Default) | Ensure as much of the element is visible as possible to showcase.               |
| None             | Does not apply any collision detection.   |

### HTML

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
</div>
</div>
// Creates the Tooltip
<script>
```

```
var sample1 = new ej.Tooltip($("#test"), {  
  content: "JavaScript is the programming language of HTML and the Web.",  
  collision : "fit"  
});  
</script>
```

## How To

### Use AJAX to generate the Tooltip's content

Use jQuery's built-in AJAX functionality to retrieve remote content and set the content to the Tooltip as follows.

Define three employee's JSON Array as given below(an array of 3 employees records)

### JS

```
[  
  {  
    "EmployeeID": 1,  
    "LastName": "Davolio",  
    "Title": "Sales Representative",  
    "TitleOfCourtesy": "Dr.",  
    "Address": "507 - 20th Ave. E.Apt. 2A",  
    "City": "Seattle",  
    "Region": "WA",  
    "PostalCode": "98122",  
  },  
  {  
    "EmployeeID": 2,  
    "LastName": "Buchanan",  
    "Title": "Sales Representative",  
    "TitleOfCourtesy": "Dr.",  
    "Address": "908 W. Capital Way",  
    "City": "Tacoma",  
    "Region": "WA",  
    "PostalCode": "98401",  
  },  
  {  
    "EmployeeID": 3,  
    "LastName": "Leverling",  
    "Title": "Sales Representative",  
    "TitleOfCourtesy": "Dr.",  
    "Address": "722 Moss Bay Blvd.",  
    "City": "Kirkland",  
    "Region": "WA",  
    "PostalCode": "98033",  
  }  
]
```

Render the employees table and create the tooltip. Once the tooltip created, the data will be fetched using AJAX method as follows.

### HTML

```
<div class="frame">  
<table id="details">
```

```

<tr>
<th>EmployeeID</th>
<th>Name</th>
<th>Designation</th>
</tr>
<tr>
<td>SF6089</td>
<td><a href="#" class="e-info" title="Davolio">Davolio</a></td>
<td>Software Engineer</td>
</tr>
<tr>
<td>SF6073</td>
<td><a href="#" class="e-info" title="Leverling">Leverling </a> </td>
<td>Tester</td>
</tr>
<tr>
<td>SF6073</td>
<td><a href="#" class="e-info" title="Suyama"> Suyama </a> </td>
<td>Content Writer</td>
</tr>
<tr>
<td>SF7896</td>
<td><a href="#" class="e-info" title="Buchanan"> Buchanan </a> </td>
<td>Graphics Designer</td>
</tr>
</table>
</div>
<script type="text/javascript">
var content;
var target = new ej.Tooltip($("#details"),{
target: ".e-info",
content: " ",
beforeOpen: "onOpen",
width : "350px",
height: "128px",
position :{
target : {horizontal : "right", vertical : "bottom"},
stem : {horizontal: "left", vertical: "top"}
}
}).data("ejTooltip");
function onOpen(args) {
proxy = args;
$.ajax({
dataType: "json",
url: "tooltipData.js",
success: function (result) {
var employee = [
{ photo:
"http://js.syncfusion.com/demos/web/content/images/grid/Employees/2.png"
},
{ photo:
"http://js.syncfusion.com/demos/web/content/images/grid/Employees/4.png"
},
{ photo:
"http://js.syncfusion.com/demos/web/content/images/grid/Employees/8.png"
},

```

```

{ photo:
"http://js.syncfusion.com/demos/web/content/images/grid/Employees/3.png"
},
{ photo:
"http://js.syncfusion.com/demos/web/content/images/grid/Employees/1.png"
},
{ photo:
"http://js.syncfusion.com/demos/web/content/images/grid/Employees/6.png"
},
];
for (i = 0; i < result.length ; i++) {
if (result[i].LastName == $(proxy.event.target).attr("data-content"))
content = '<div class="main"> <img src=' + employee[i].photo + '
class="logo"/><div class="des"><table> <tr> <th> ' +
result[i].TitleOfCourtesy + ' ' + result[i].LastName + '</th> </tr> <tr>
<td> Title </td> <td>#160;#160;: ' + result[i].Title + '</td> </tr>
<tr><td> Address</td><td>#160;#160;: ' + result[i].Address + '</td>
</tr><tr> <td> City </td> <td>#160;#160;: ' + result[i].City + '</td>
</tr> <tr> <td> PostalCode </td> <td>#160;#160;: ' +
result[i].PostalCode + '</td> </tr> </table></div></div>';
}
target.setModel({ content: content });
}
});
target.show(args.event.target);
args.cancel = true;
}
</script>

```

Defines the style for the tooltip layout and table as follows.

### HTML

```

<style>
.frame {
width: 100%;
box-sizing: border-box;
}
.frame table {
border-collapse: collapse;
width: 100%;
}
.frame th, .frame td {
text-align: left;
padding: 8px;
}
.logo {
float: left;
width: 100px;
height: 114px;
}
.frame tr:nth-child(even) {
background-color: #f2f2f2;
}
.frame th {
background-color: #4CAF50;


```

```

color: white;
}
.des {
width: 230px;
float: right;
line-height: 24px;
}
</style>

```

| EmployeeID | Name      | Designation          |
|------------|-----------|----------------------|
| SF6089     | Davolio   | Sales Representative |
| SF6073     | Leverling | Sales Representative |
| SF7896     | Buchanan  |                      |



**Dr. Buchanan**  
Title : Sales Representative  
Address : 908 W. Capital Way  
City : Tacoma  
PostalCode : 98401

### Integration with the Slider control

Tooltip can also be integrated with various other jQuery plugin. Tooltip shows the slider value above the handle.

Render the slider control and finds its handle to render the Tooltip as follows

### HTML

```

<div class="frame">
<div id="loanheading">
Details of Loan
</div>
<span class="ColumnLeft">
<span class="loan">Loan Amount</span>
</span>
<span class="ColumnRight">
<span class="value">1000</span><span id="loantext">$ </span>
</span>
<div id="loanSlider">
</div>
</div>
<script>
$(function () {
var loan = 100;
var sample1 = new ej.Slider($("#loanSlider"), {
height: 16,
value: loan,
showTooltip : false,
minValue: 10,
maxValue: 1000,
incrementStep: 10,

```

```

change: "onSlide",
slide: "onSlide",
stop : "onStop"
});
//Create the Tooltip
var loadTip = $($('#loanSlider').find(".e-handle")).ejTooltip({
content: "$ " + loanObj.getValue().toString(),
showRoundedCorner: true,
isBalloon : false
}).data("ejTooltip");
});
</script>

```

Once the handler in the slider, start its sliding the below function will be triggered

### HTML

```

function onSlide(args) {
this.wrapper.prev().children('span.value').html(args.value);
var target = "#" + args.id, value = args.value.toString();
if(args.id == "loanSlider")
value = "$ " +value;
var tipWidth = (value < 10) ? "13px" : (10 <= value >= 100) ? "25px" :
"auto";
var tipObj = $($("div" + target).find(".e-handle")).data("ejTooltip");
tipObj.setModel({content : value, width : tipWidth});
tipObj.show($("div" + target).find(".e-handle"));
}

```

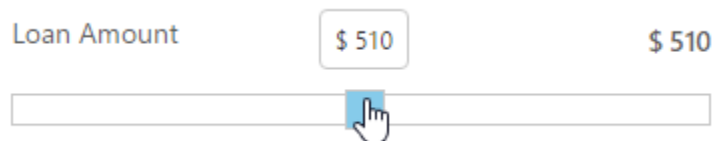
Once the sliding is stopped, the Tooltip will be shown for the particular period of time then it will be closed.

### HTML

```

function onStop(args){
setTimeout( function(){
var target = "#" + args.id;
var tipObj = $($("div" + target).find(".e-handle")).data("ejTooltip");
tipObj.hide();
}, 1000);
}

```



### Tip(arrow) customization

Styling the Tip's background and border colors is done using "cssClass" API of Tooltip. Change the size of the tip using the property name called "Tip" and CSS as follows.

### HTML

```
<div class="frame">
```

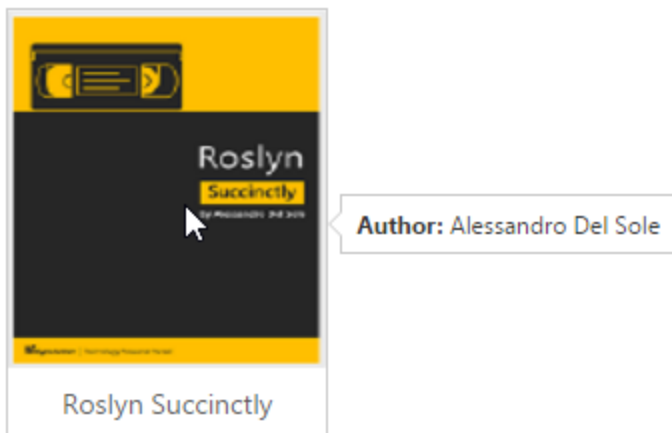
```
<div class="img" id="custom">

<div class="desc">Roslyn Succinctly</div>
</div>
</div>
<script type="text/javascript">
$(function () {
target = new ej.Tooltip($("#custom"),{
content : "Learn to use the Python language to create programs of all
kinds to creating practical applications.",
width: "200px",
tip :{
size : { width : 25, height : 12}
},
position :{
stem: { horizontal: "left", vertical: "center" },
target: { horizontal : "right", vertical : "center"}
}
}).data("ejTooltip");
});
</script>
```

Defines the style for the tip as follow as

#### HTML

```
<style>
div.img {
border: 1px solid #ccc;
width: 159px;
height: 213px;
left: 35%;
position: relative;
top: 20%;
}
div.img img{
width: 159px;
height: 179px;
}
div.desc {
padding: 8px;
text-align: center;
}
</style>
```



### Initialize Tooltip for the target element

Tooltips are also useful for form elements, to show some additional information in the context of each field.

Rather than using multiple Tooltips, a single Tooltip can be used to show the information of each and every field of the form/table element. This can be achieved using “target” API. Target property Specifies a selector for elements, within the container, for which the Tooltip will be displayed.

The example below demonstrates how to create a Tooltip for multiple targets element.

### HTML

```
<form id="details" role="form">
  <fieldset>
    <table>
      <tr>
        <td class="info"> User Name:</td>
        <td> <input type="text" class="e-info" name="firstname" title="Please
enter your name"> </td>
      </tr>
      <tr>
        <td class="info"> Email Address:</td>
        <td> <input type="text" class="e-info" name="email" title="Enter a valid
email address"></td>
      </tr>
      <tr>
        <td class="info"> Password:</td>
        <td> <input type="password" class="e-info" name="password" title="Be at
least 8 characters length"></td>
      </tr>
      <tr>
        <td class="info"> Confirm Password:</td>
        <td> <input type="password" class="e-info" name="Cpwd" title="Re-enter
your password"></td>
      </tr>
    </table>
    <input id="sample" type="submit" value="Submit">
    <input id="clear" type="reset" value="Reset">
  </fieldset>
</form>
<script type="text/javascript">
```



```

var sample1 = new ej.Tooltip($("#details"), {
width: "180px",
height: "30px",
position: {
stem: { horizontal: "left", vertical: "center"},
target:{ horizontal: "right", vertical: "center"}
},
target: ".e-info"
});
var buttoninstance = new ej.Button($("#sample"), {
showRoundedCorner: true,
size: "medium"
});
var buttoninstance1 = new ej.Button($("#clear"), {
showRoundedCorner: true,
size: "medium"
});
</script>

```

The image shows a web form for user registration. It contains four text input fields labeled 'User Name:', 'Email Address:', 'Password:', and 'Confirm Password:'. The 'Email Address:' field is currently active, indicated by a blue border and a text cursor. A tooltip box is positioned to the right of this field, containing the text 'Enter a valid email address'. Below the input fields are two buttons: 'Submit' and 'Reset'.

Apply the following styles to the form element.

#### HTML

```

<style>
th, td {
padding: 15px;
text-align: left;
}
#form {
position: absolute;
left: 200px;
}
fieldset {
border: 1px solid #D9DEDD;
padding: 15px;

```

```
width: 400px;
}
.info {
font-weight: bold;
}
#sample {
margin-right: 50px;
}
.e-button {
margin: 10px;
}
</style>
```

### Interact with the Tooltip

Give users, the possibility to interact with the tooltip. If the tooltip is interactive and activated by a hover event, set the amount of time (milliseconds) allowed for a user to hover off of the tooltip activator on to the tooltip itself – keeping the tooltip from closing using “autoCloseTimeout” property.

Using this property, Links can be provided in Tooltip content where user can navigate to some other page. This is handy for situations where you need to grab the user attention.

### HTML

```
<div class="control">
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
</div>
// Creates the Tooltip
<script>
var sample1 = new ej.Tooltip($("#sample"), {
content: "JavaScript is the programming language of HTML and the Web.",
closeMode : "auto",
autoCloseTimeout: 2000
});
</script>
```

## TreeGrid

### Overview

**The Essential TypeScript TreeGrid** is an efficient control designed for representing the hierarchical data in a tabular format, combining the visual representation of Grid and TreeView controls; it represents the data from datasource such as array of JSON objects, ej.DataManager or self-referential datasource.

### Key Features

- **Editing** - Offers cell editing Mode for editing the item along each column
- **Sorting** - Supports  $n$  levels of sorting.
- **Column Template** - Offers to render the customized column for an item also with customized expand-collapse icon.
- **Virtualization** - Supports rendering huge amount of hierarchical data at once.
- **User Interaction** - Supports tooltip for each cell or even only for expander cell in the grid, expand collapse at ease, single and multiple row selection, columns resizing.

## Getting Started

This section helps to understand the getting started of the TypeScript TreeGrid with the step-by-step instructions.

### Create your first TreeGrid in TypeScript

To get started Syncfusion TypeScript application refer [this](#) page for basic control integration and script references.

The **Essential TypeScript TreeGrid** has been designed to represent and edit the hierarchical data.

This section explains how to create a TreeGrid widget in your application with hierarchical data source and enable sorting and editing. The following screenshot displays the output.

| Task Id | Task Name                  | Start Date | End Date   | Progress |
|---------|----------------------------|------------|------------|----------|
| 1       | ▲ Planning                 | 02/03/2014 | 02/07/2014 | 100      |
| 2       | Plan timeline              | 02/03/2014 | 02/07/2014 | 100      |
| 3       | Plan budget                | 02/03/2014 | 02/07/2014 | 100      |
| 4       | Allocate resources         | 02/03/2014 | 02/07/2014 | 100      |
| 5       | Planning complete          | 02/07/2014 | 02/07/2014 | 0        |
| 6       | ▲ Design                   | 02/10/2014 | 02/14/2014 | 86       |
| 7       | Software Specification     | 02/10/2014 | 02/12/2014 | 60       |
| 8       | Develop prototype          | 02/10/2014 | 02/12/2014 | 100      |
| 9       | Get approval from customer | 02/13/2014 | 02/14/2014 | 100      |
| 10      | Design Documentation       | 02/13/2014 | 02/14/2014 | 100      |
| 11      | Design complete            | 02/14/2014 | 02/14/2014 | 0        |

It is necessary to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the jquery.d.ts file in your TypeScript application, which can be downloaded from [here](#).

1. Create HTML file and add the following necessary script and css files to the HTML file.

### HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
<meta charset="utf-8" />
<link href="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet"/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-1.10.2.min.js"></script>
```

```
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"></scri
pt>
<script src="http://cdn.syncfusion.com/{ site.releaseversion
}/js/web/ej.web.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add TreeGrid control here -->
</body>
</html>
```

2.Add <div> element with in the <Body> tag.

### HTML

```
<div class="cols-sample-area">
<div id="TreeGridContainer"></div>
</div>
<script type="text/javascript" src="treegrid/treegrid.js"></script>
```

3.Create the TreeGrid with the empty data source.

### JS

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module TreeGridComponent {
$(function() {
var treeGridInstance = new ej.TreeGrid($("#TreeGridContainer"), {
columns: [
{ field: "taskID", headerText: "Task Id", allowFiltering: false },
{ field: "taskName", headerText: "Task Name" },
{ field: "startDate", headerText: "Start Date" },
{ field: "endDate", headerText: "End Date" },
{ field: "progress", headerText: "Progress" }],
});
});
}
```

| Task Id               | Task Name | Start Date | End Date | Progress |
|-----------------------|-----------|------------|----------|----------|
| No records to display |           |            |          |          |

TreeGrid with empty datasource

4.Create data source for TreeGrid.

### JS

```
var treeGridDataSource = [{
taskID: 2,
taskName: "Planning",
startDate: "02/03/2014",
```

```
endDate: "02/07/2014",
duration: 10,
progress: 56,
subtasks: [{
  taskID: 3,
  taskName: "Plan timeline",
  startDate: "02/03/2014",
  endDate: "02/07/2014",
  duration: 5,
  progress: "100"
}, {
  taskID: 4,
  taskName: "Plan budget",
  startDate: "02/03/2014",
  endDate: "02/07/2014",
  duration: 5,
  progress: "100"
}, {
  taskID: 5,
  taskName: "Allocate resources",
  startDate: "02/03/2014",
  endDate: "02/07/2014",
  duration: 5,
  progress: "100"
}, {
  taskID: 6,
  taskName: "Planning complete",
  startDate: "02/07/2014",
  endDate: "02/07/2014",
  duration: 0,
  progress: 40
}]
}, {
  taskID: 7,
  taskName: "Design",
  startDate: "02/10/2014",
  endDate: "02/14/2014",
  duration: 10,
  progress: 76,
  subtasks: [{
    taskID: 8,
    taskName: "Software Specification",
    startDate: "02/10/2014",
    endDate: "02/12/2014",
    duration: 3,
    progress: "60"
  }, {
    taskID: 9,
    taskName: "Develop prototype",
    startDate: "02/10/2014",
    endDate: "02/12/2014",
    duration: 3,
    progress: "100"
  }, {
    taskID: 10,
    taskName: "Get approval from customer",
    startDate: "02/13/2014",
```

```
endDate: "02/14/2014",
duration: 2,
progress: "100"
}, {
  taskID: 11,
  taskName: "Design complete",
  startDate: "02/14/2014",
  endDate: "02/14/2014",
  duration: 0,
  predecessor: "10FF",
  progress: 65
}]
}
];
```

5. Initialize the TreeGrid with data source created in last step.

### JS

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module TreeGridComponent {
  $(function() {
    var treeGridInstance = new ej.TreeGrid($("#TreeGridContainer"), {
      dataSource: (<any> window).treeGridDataSource,
      childMapping: "subtasks",
      columns: [
        { field: "taskID", headerText: "Task Id", allowFiltering: false },
        { field: "taskName", headerText: "Task Name" },
        { field: "startDate", headerText: "Start Date" },
        { field: "endDate", headerText: "End Date" },
        { field: "progress", headerText: "Progress" }],
    });
  });
}
```

TreeGrid widget is displayed as the output in the following screenshot.

| Task Id | Task Name                  | Start Date | End Date   | Progress |
|---------|----------------------------|------------|------------|----------|
| 1       | ▲ Planning                 | 02/03/2014 | 02/07/2014 | 100      |
| 2       | Plan timeline              | 02/03/2014 | 02/07/2014 | 100      |
| 3       | Plan budget                | 02/03/2014 | 02/07/2014 | 100      |
| 4       | Allocate resources         | 02/03/2014 | 02/07/2014 | 100      |
| 5       | Planning complete          | 02/07/2014 | 02/07/2014 | 0        |
| 6       | ▲ Design                   | 02/10/2014 | 02/14/2014 | 86       |
| 7       | Software Specification     | 02/10/2014 | 02/12/2014 | 60       |
| 8       | Develop prototype          | 02/10/2014 | 02/12/2014 | 100      |
| 9       | Get approval from customer | 02/13/2014 | 02/14/2014 | 100      |
| 10      | Design Documentation       | 02/13/2014 | 02/14/2014 | 100      |
| 11      | Design complete            | 02/14/2014 | 02/14/2014 | 0        |

### Enable Sorting

The TreeGrid control has sorting functionality, to arrange the data in ascending or descending order based on a particular column.

### Multicolumn Sorting




Enable the multicolumn sorting in TreeGrid by setting `allowMultiSorting` as `true`. You can sort multiple columns in TreeGrid, by selecting the desired column header while holding the **Ctrl** key.

### JS

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module TreeGridComponent {
    $(function() {
        var treeGridInstance = new ej.TreeGrid($("#TreeGridContainer"), {
            allowSorting: true,
            allowMultiSorting: true
        });
    });
}

```

| Task Id | Task Name   | Start Date | End Date  | Progress |
|---------|---|------------|--|----------|
| 1       | Planning  | 02/03/2014 | 02/07/2014   | 100      |
| 4       | Allocate resources  | 02/03/2014 | 02/07/2014   | 100      |
| 3       | Plan budget   | 02/03/2014 | 02/07/2014   | 100      |
| 2       | Plan timeline   | 02/03/2014 | 02/07/2014   | 100      |
| 5       | Planning complete   | 02/07/2014 | 02/07/2014   | 0        |
| 6       | Design  | 02/10/2014 | 02/14/2014   | 86       |
| 8       | Develop prototype   | 02/10/2014 | 02/12/2014   | 100      |
| 7       | Software Specification  | 02/10/2014 | 02/12/2014   | 60       |
| 11      | Design complete   | 02/14/2014 | 02/14/2014   | 0        |
| 10      | Design Documentation  | 02/13/2014 | 02/14/2014   | 100      |
| 9       | Get approval from customer  | 02/13/2014 | 02/14/2014   | 100      |

### Enable Editing

You can enable Editing in TreeGrid by using the [editSettings](#) property as follows.

### JS

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module TreeGridComponent {
    $(function() {
        var treeGridInstance = new ej.TreeGrid($("#TreeGridContainer"), {
            editSettings: {
                allowAdding: true,
                allowEditing: true,
                allowDeleting: true,
                editMode: "cellEditing",
                rowPosition: "belowSelectedRow"
            },
        });
    });
}

```

And also, the following editors are provided for support in TreeGrid control.

- string
- boolean
- numeric
- dropdown
- datepicker
- dateTimePicker

You can set the editor type for a particular column as follows.

### JS



```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module TreeGridComponent {
$(function() {
var treeGridInstance = new ej.TreeGrid($("#TreeGridContainer"), {
columns: [
{ field: "taskId", headerText: "Task Id", allowFiltering: false,
editType: ej.TreeGrid.EditingType.Numeric },
{ field: "taskName", headerText: "Task Name", editType:
ej.TreeGrid.EditingType.String },
{ field: "startDate", headerText: "Start Date", editType:
ej.TreeGrid.EditingType.DatePicker },
{ field: "endDate", headerText: "End Date", editType:
ej.TreeGrid.EditingType.DatePicker },
{ field: "progress", headerText: "Progress", editType:
ej.TreeGrid.EditingType.Numeric }
],
});
});
}

```

The output of the DateTimePicker editor in TreeGrid control is as follows.

| Task Id | Task Name                  | Start Date           | End Date   | Progress |
|---------|----------------------------|----------------------|------------|----------|
| 1       | Planning                   | 02/03/2014           | 02/07/2014 | 100      |
| 2       | Plan timeline              | 02/03/2014           | 02/07/2014 | 100      |
| 3       | Plan budget                | 02/03/2014           | 02/07/2014 | 100      |
| 4       | Allocate resources         | February 2014        |            | 100      |
| 5       | Planning complete          | Su Mo Tu We Th Fr Sa |            | 0        |
| 6       | Design                     | 26 27 28 29 30 31 1  |            | 86       |
| 7       | Software Specification     | 2 3 4 5 6 7 8        |            | 60       |
| 8       | Develop prototype          | 9 10 11 12 13 14 15  |            | 100      |
| 9       | Get approval from customer | 16 17 18 19 20 21 22 |            | 100      |
| 10      | Design Documentation       | 23 24 25 26 27 28 1  |            | 100      |
| 11      | Design complete            | Today                |            | 0        |

## TreeMap

### Overview

**Essential TreeMap** for JavaScript is ideal for visualizing large amounts of data. **TreeMap** holds a set of nested nodes to display hierarchical data. Each nested node may contain sub-nodes containing area proportional to a specified data value bounded. Color and annotation can be used to provide extra information about the leaf nodes. **EssentialTreeMap** is a perfect solution for developers looking to add advanced, feature rich **Treemap** to their applications.

## Key Features

- **Levels - TreeMap** levels are used to define levels of various flat data and hierarchical data collection.
- **Layout** - Layouts such as Squarified, SliceAndDiceAuto, SliceAndDiceHorizontal and SliceAndDiceVertical determines the visual representation of nodes belonging to all the treemap levels in TreeMap.
- **Visualization for Colors** - Easy customization options is provided to customize the colors of the leaf nodes of **TreeMap**.

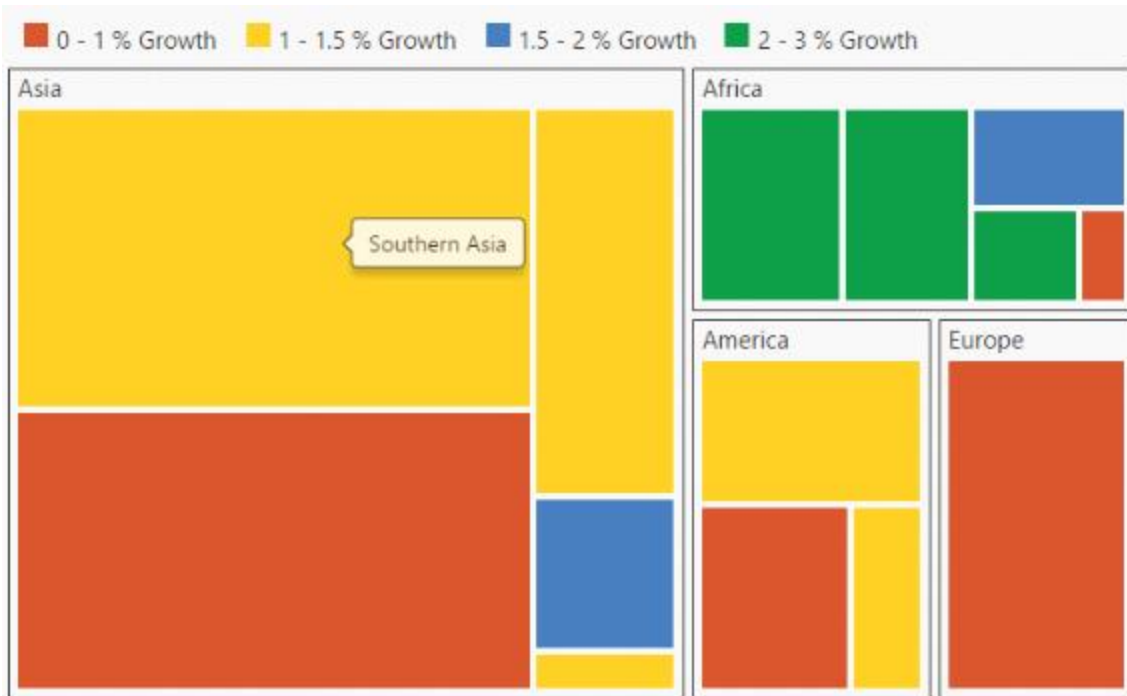
## Use Case Scenarios

TreeMap is used to represent large or complex data sets in various applications, such as:

- Stock market analysis where the weight of each stock in the index is represented by the size of the rectangle and its range of loss or gain is represented by the color of the rectangle.
- Visualizing Internet usage in certain categories, such as retail, social networks, and search.
- Categorizing the news aggregated by Google News where the colors represent different sections, such as business or politics, and the size of the boxes represent how many similar stories also appear in Google News.
- Indicating weather report analysis around the world. The opacity of each rectangle can differ based on its humidity.

## Getting Started

- This section explains briefly about how to create a TreeMap in your application with TypeScript.
- Here you can learn how to configure a TreeMap control in a real-time scenario where it is used to visually represent the percentage of growth in population in each continent.
- It also provides a walk-through on some of the customization features available in TreeMap control.



### Create a TreeMap

You can easily create the TreeMap widget by using the following steps.

#### Add Libraries

1. First create an TypeScript Project and add the following script reference in the app.ts page

For common getting started of TypeScript, you can refer [here](#).

The default type definition file **ej.web.all.d.ts** needs to include the support for type-checking while initializing any of the Syncfusion widgets.

The important step you need to do is to copy the ej.web.all.d.ts file into your project and then need to refer it in your TypeScript application (app.ts file), so that you will get the IntelliSense support and also the compile time type-checking.

You can find the ej.web.all.d.ts file in the following location,

(installed location) \Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\typescript

Apart from ej.web.all.d.ts file, it is also necessary to make use of the **jquery.d.ts** file in your TypeScript application, which can be downloaded from [here](#).

2. Add the following script reference in the HTML page

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
```

```
<script src="http://cdn.jsdelivr.net/jsrender/1.0pre35/jsrender.min.js"
type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

In the above code, `ej.web.all.min.js` script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use [CSG](#) utility to generate a custom script file with the required widgets for deployment purpose.

### Initialize TreeMap

#### 1.Create a

tag with specific id

#### HTML

```
<html> <body> <div id="TreeMap"></div> </body> </html>
```

2.The `dataSource` property of the TreeMap accepts the collection values as input.Populate the datasource of the TreeMap data as JSON object. For example, you can use population data of countries to generate TreeMap data as illustrated in the following code sample.

#### HTML

```
var population_data = [
{ Continent: "Asia", Region: "Southern Asia", Growth: 1.32, Population:
1749046000 },
{ Continent: "Asia", Region: "Eastern Asia", Growth: 0.57, Population:
1620807000 },
{ Continent: "Asia", Region: "South-Eastern Asia", Growth: 1.20,
Population: 618793000 },
{ Continent: "Asia", Region: "Western Asia", Growth: 1.98, Population:
245707000 },
{ Continent: "Asia", Region: "Central Asia", Growth: 1.43, Population:
64370000 },
{ Continent: "Europe", Region: "Europe", Growth: 0.10, Population:
742452000 },
{ Continent: "America", Region: "South America", Growth: 1.06,
Population: 406740000 },
{ Continent: "America", Region: "Northern America", Growth: 0.85,
Population: 355361000 },
{ Continent: "America", Region: "Central America", Growth: 1.40,
Population: 167387000 },
{ Continent: "Africa", Region: "Eastern Africa", Growth: 2.89,
Population: 373202000 },
{ Continent: "Africa", Region: "Western Africa", Growth: 2.78,
Population: 331255000 },
{ Continent: "Africa", Region: "Northern Africa", Growth: 1.70,
Population: 210002000 },
{ Continent: "Africa", Region: "Middle Africa", Growth: 2.79, Population:
135750000 },
```

```
{ Continent: "Africa", Region: "Southern Africa", Growth: 0.91,  
Population: 60425000 }  
];
```

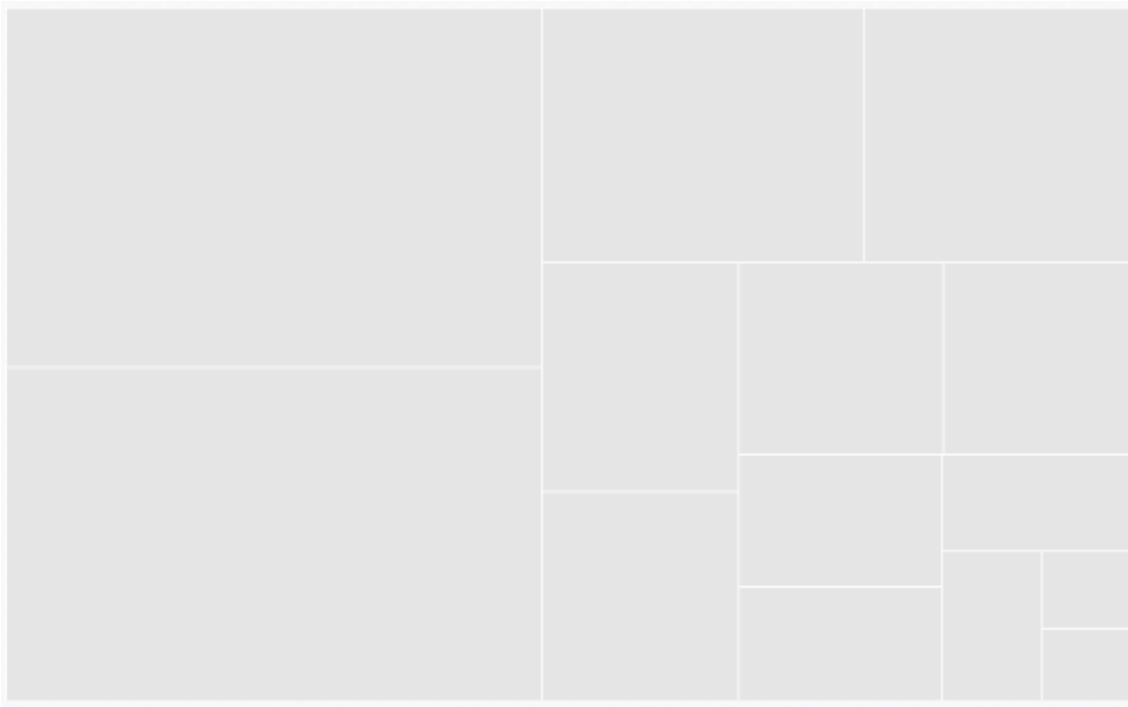
**Note:** Population data is referred from [List of continents by population](#)

3.The size of an object can be calculated by using the **WeightValuePath** of TreeMap 4.Initialize the TreeMap in ts file by using the `ej.TreeMap` method.

### JAVASCRIPT

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module TreeMapComponent {  
    $(function () {  
        var treeMapSample = new ej.datavisualization.TreeMap($("#TreeMap"), {  
            dataSource: population_data,  
            weightValuePath: "Population",  
        });  
    });  
}
```

The following image displays a TreeMap with default properties using the above code.



### GroupTreeMap Items using Levels

You can group TreeMap Items using levels in TreeMap.

#### Group Path

You can use `groupPath` property for every flat level of the TreeMap control. It is a path to a field on the source object that serves as the "Group" for the level specified. You can group the data based on the

`groupPath` in the TreeMap control. When the `groupPath` is not specified, then the items are not grouped and the data is displayed in the order specified in the `dataSource`.

#### Group Gap

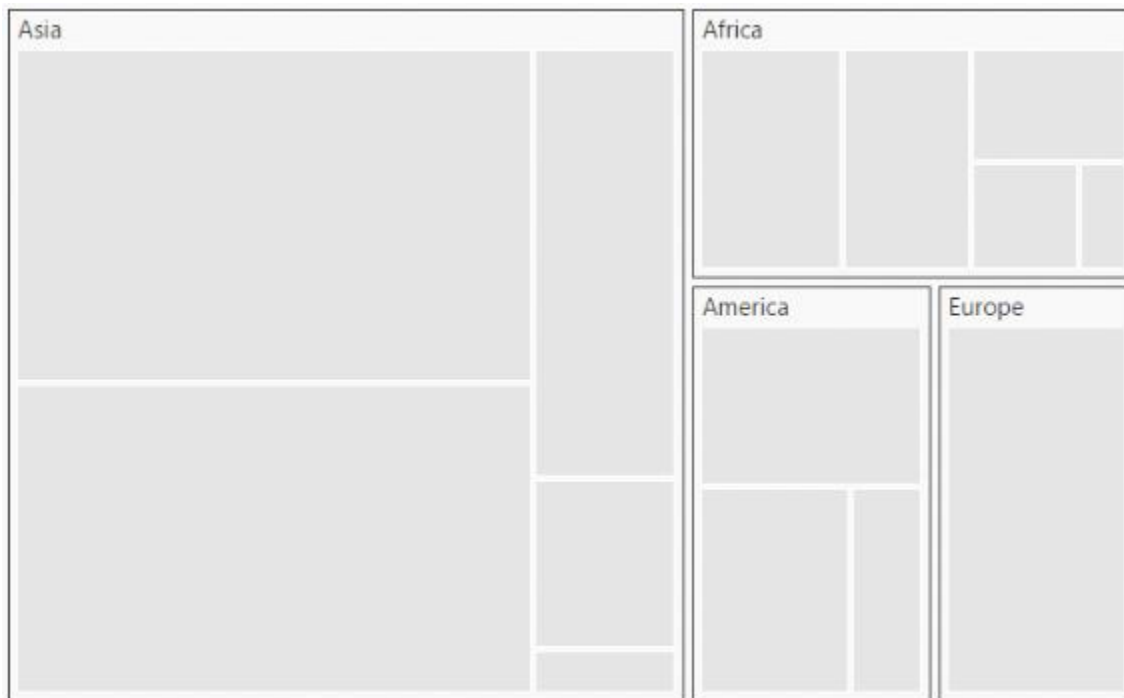
You can use `groupGap` property to separate the items from every flat level and to differentiate the levels mentioned in the TreeMap control.

The following code sample explains how to group TreeMap Items using 'Levels'

#### JAVASCRIPT

```
$(function () {  
    var treeMapSample = new ej.datavisualization.TreeMap($("#TreeMap"), {  
        dataSource: population_data,  
        weightValuePath: "Population",  
        levels: [  
            { groupPath: "Continent", groupGap: 5}  
        ]  
    });  
});
```

The following screenshot displays grouping of TreeMapItems using Levels.



#### Customize TreeMap Appearance by Range

You can differentiate the nodes based on its value and color ranges using Range color. You can also define the color value range using From and To properties.

#### Color Value Path

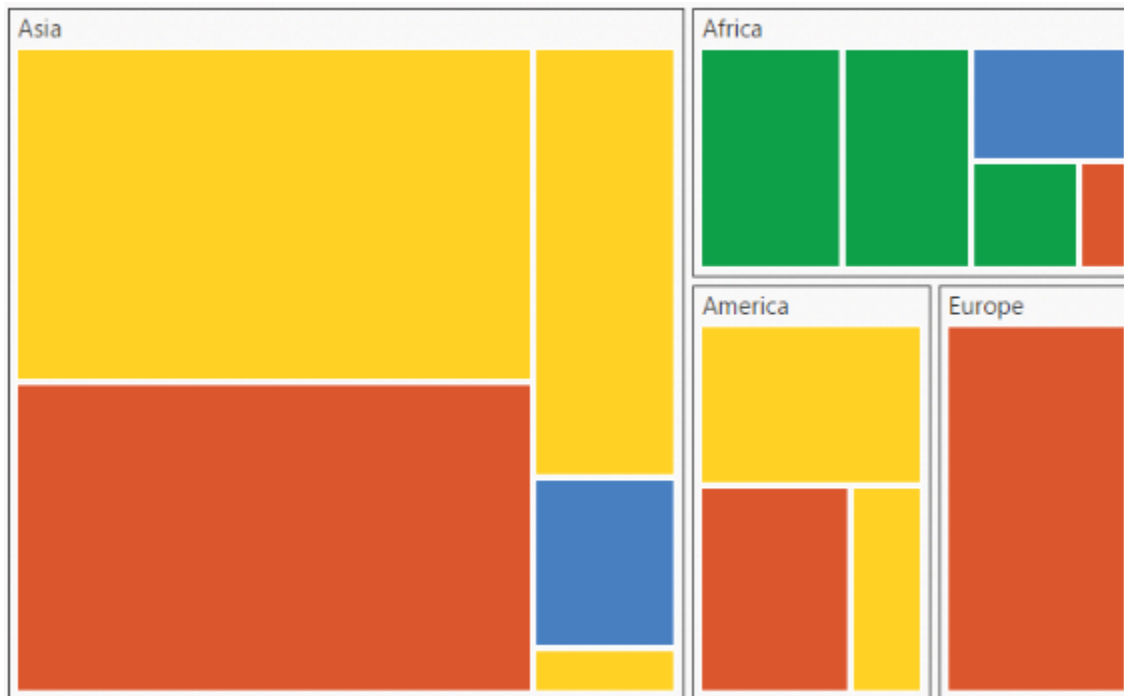
The `colorValuePath` of TreeMap is a path to a field on the source object. You can determine the color for the object using `colorValuePath` of TreeMap.

The following code sample explains how to customize TreeMap Appearance by Range.

## JAVASCRIPT

```
$(function () {
var treeMapSample = new ej.datavisualization.TreeMap($("#TreeMap"), {
dataSource: population_data,
weightValuePath: "Population",
levels: [
{ groupPath: "Continent", groupGap: 5 }
],
colorValuePath: "Growth",
rangeColorMapping: [
{ color: "#DC562D", from: "0", to: "1" },
{ color: "#FED124", from: "1", to: "1.5" },
{ color: "#487FC1", from: "1.5", to: "2" },
{ color: "#0E9F49", from: "2", to: "3" }
]
});
});
```

The following screenshot displays customized TreeMap Appearance by Range



### Enable Tooltip

You can enable the tooltip by setting `showTooltip` property to 'true'. By default, it takes the property of the bound object that is referred in the `weightValuePath` and displays its content when the corresponding node is hovered. You can customize the template for tooltip using `tooltipTemplate` property.

### Leaf Item Settings

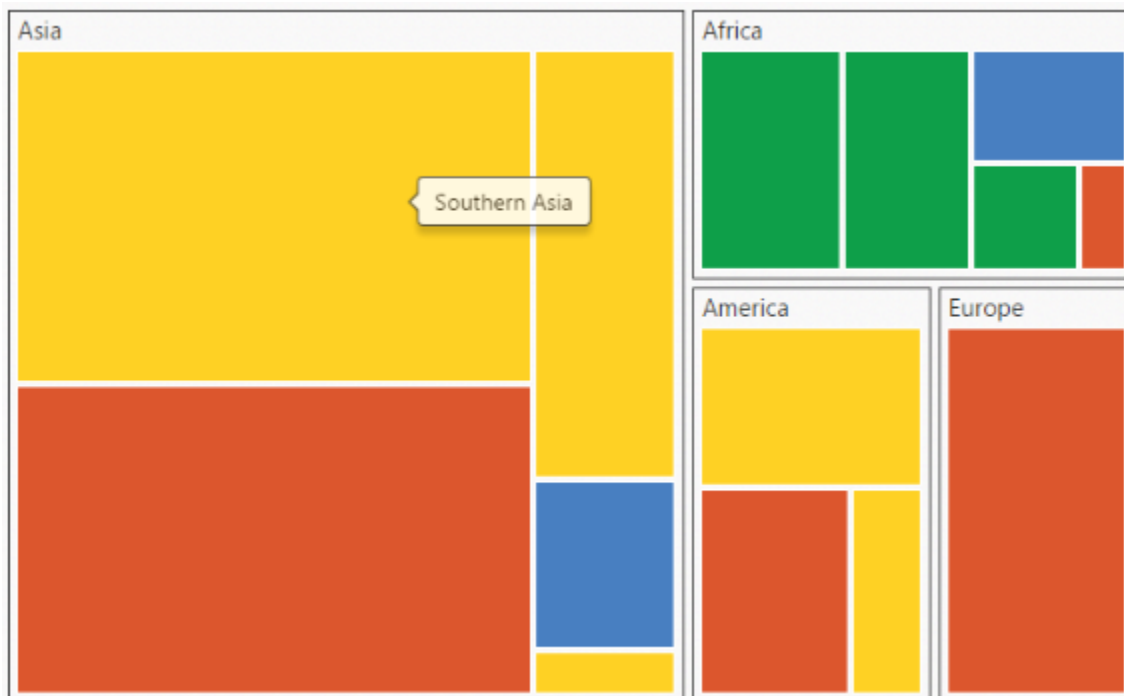
You can customize the Leaf level TreeMap items using `leafItemSettings`. The Label and tooltip values take the property of bound object that is referred in the `labelPath` when defined.

The following code sample displays how the tooltip is enabled.

### JAVASCRIPT

```
$(function () {  
  var treeMapSample = new ej.datavisualization.TreeMap($("#TreeMap"), {  
    dataSource: population_data,  
    weightValuePath: "Population",  
    levels: [  
      { groupPath: "Continent", groupGap: 5}  
    ],  
    colorValuePath: "Growth",  
    rangeColorMapping: [  
      { color: "#DC562D", from: "0", to: "1" },  
      { color: "#FED124", from: "1", to: "1.5" },  
      { color: "#487FC1", from: "1.5", to: "2" },  
      { color: "#0E9F49", from: "2", to: "3" }  
    ],  
    showTooltip: true,  
    leafItemSettings: { labelPath: "Region" }  
  });  
});
```

The following screenshot displays the TreeMap when the Tooltip is enabled.



### Legend

You can set the color value of leaf nodes using TreeMap Legend. This legend is appropriate only for the TreeMap whose leaf nodes are colored using `rangeColorMapping`.

You can set ShowLegend property value to 'true' to make a Legend visible.



### Label for Legend

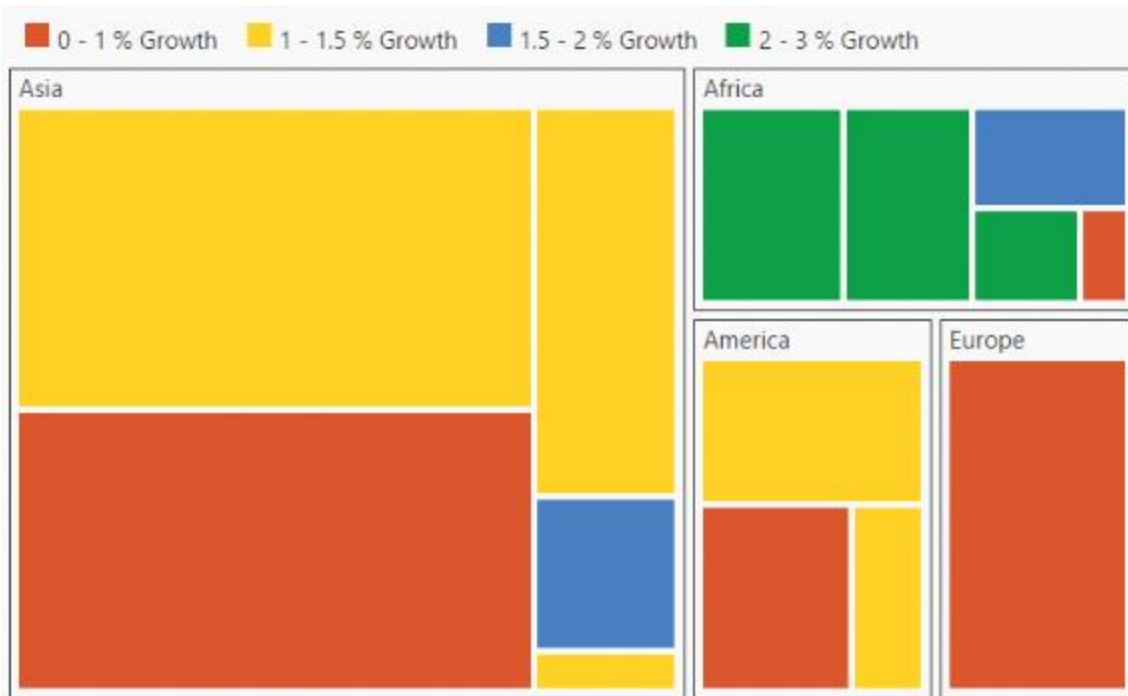
You can customize the labels of the legend item using `legendLabel` property of `rangeColorMapping`.

The following code sample displays how to add labels for legend in a TreeMap.

#### JAVASCRIPT

```
$(function () {  
  var treeMapSample = new ej.datavisualization.TreeMap($("#TreeMap"), {  
    dataSource: population_data,  
    weightValuePath: "Population",  
    levels: [  
      { groupPath: "Continent", groupGap: 5}  
    ],  
    colorValuePath: "Growth",  
    rangeColorMapping: [  
      { color: "#DC562D", from: "0", to: "1", **legendLabel: "0 - 1 %  
Growth** },  
      { color: "#FED124", from: "1", to: "1.5", **legendLabel: "1 - 1.5 %  
Growth** },  
      { color: "#487FC1", from: "1.5", to: "2", **legendLabel: "1.5 - 2 %  
Growth** },  
      { color: "#0E9F49", from: "2", to: "3", **legendLabel: "2 - 3 % Growth**  
      }  
    ],  
    showTooltip: true,  
    leafItemSettings: { labelPath: "Region" },  
    showLegend: true,  
    legendSettings: {  
      showLegend: true,  
      height: 38,  
      width: 690,  
    }  
  });  
});
```

The following screenshot displays the TreeMap when Labels are enabled.



## DataBinding

**TreeMap** control supports Data Binding and it can be achieved using `dataSource` property.

The `dataSource` property accepts the collection values as input. For example, you can provide the list of objects as input. The following code illustrates you on how to bind a flat collection as datasource for **TreeMap**.

## JAVASCRIPT

```

/// <reference path="../../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../../tsfiles/ej.web.all.d.ts"></reference>
module treeMapcomponent {
  $(function () {
    var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {
      dataSource: population_data,
      colorValuePath: "Growth",
      weightValuePath: "Population"
    });
  });
}

var population_data = [
  { Continent: "Asia", Region: "Southern Asia", Growth: 1.32, Population: 1749046000 },
  { Continent: "Asia", Region: "Eastern Asia", Growth: 0.57, Population: 1620807000 },
  { Continent: "Asia", Region: "South-Eastern Asia", Growth: 1.20, Population: 618793000 },
  { Continent: "Asia", Region: "Western Asia", Growth: 1.98, Population: 245707000 },
  { Continent: "Asia", Region: "Central Asia", Growth: 1.43, Population: 64370000 },
  { Continent: "Europe", Region: "Europe", Growth: 0.10, Population: 742452000 },
];

```

```
{ Continent: "America", Region: "South America", Growth: 1.06,
Population: 406740000 },
{ Continent: "America", Region: "Northern America", Growth: 0.85,
Population: 355361000 },
{ Continent: "America", Region: "Central America", Growth: 1.40,
Population: 167387000 },
{ Continent: "Africa", Region: "Eastern Africa", Growth: 2.89,
Population: 373202000 },
{ Continent: "Africa", Region: "Western Africa", Growth: 2.78,
Population: 331255000 },
{ Continent: "Africa", Region: "Northern Africa", Growth: 1.70,
Population: 210002000 },
{ Continent: "Africa", Region: "Middle Africa", Growth: 2.79, Population:
135750000 },
{ Continent: "Africa", Region: "Southern Africa", Growth: 0.91,
Population: 60425000 }
];
```

## TreeMapLevels

The **levels** of **TreeMap** can be categorized into two types as,

- FlatLevel
- Hierarchical Level

Following customization options are available to customize the treemap level as per your requirements.

- To specify the background color for the group, you can use **groupBackground** property.
- To specify the border color for the group, you can use **groupBorderColor** property.
- To maintain the border thickness for the group, you can use **groupBorderThickness** property.
- You can specify the gaps between groups using **groupGap** property.
- You can specify the padding using **groupPadding** property.
- For specifying the header height, you can use **headerHeight** property.
- You can customize the header template using **headerTemplate** property.
- To specify the label position, you can use **labelPosition** property.
- To specify the label template for treemap, you can use **labelTemplate** property.
- You can specify the label visibility using **labelVisibilityMode** property.
- You can control the label visibility using **showLabels** property.
- For controlling text overflow, you can use **textOverflow** property.

## Flat Level

### GroupPath

You can use **groupPath** property for every flat level of the **TreeMap** control. It is a path to a field on the source object that serves as the “**Group**” for the level specified. You can group the data based on the **groupPath** in the **TreeMap** control. When the **groupPath** is not specified, then the items are not grouped and the data is displayed in the order specified in the **dataSource**.

### GroupGap

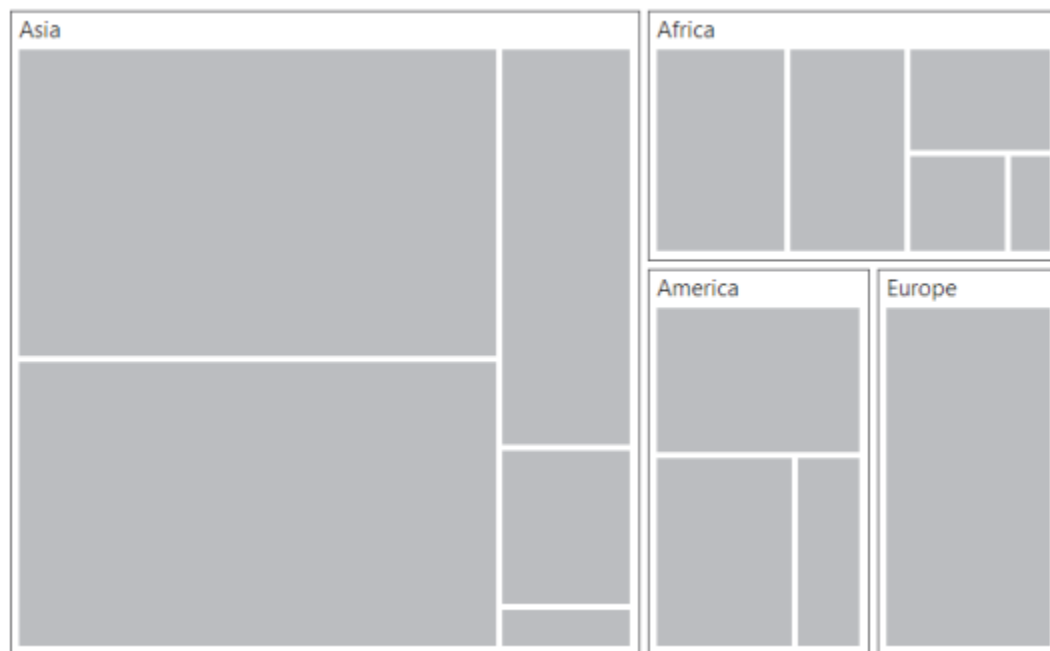
You can use `groupGap` property to separate the items from every flat level and to differentiate the levels mentioned in the **TreeMap** control.

### JAVASCRIPT

```

/// <reference path="../../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../../tsfiles/ej.web.all.d.ts"></reference>
module treeMapcomponent {
  $(function () {
    var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {
      dataSource: population_data,
      colorValuePath: "Growth",
      weightValuePath: "Population",
      levels: [
        { groupPath: "Continent", groupGap: 5}
      ]
    });
  });
}

```



Try it: [FlatLevel](#)

### Hierarchical Level

**TreeMap** Hierarchical level is used to define levels for hierarchical data collection that contains tree-structured data.

### JAVASCRIPT

```

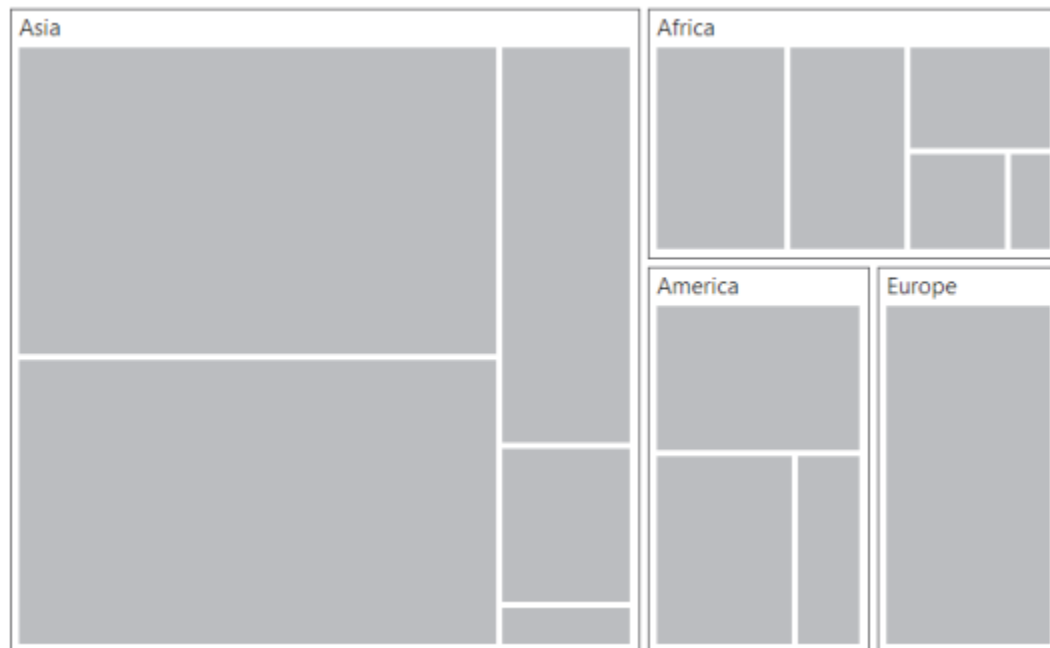
$(function ($) {
  var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {
    dataSource: population_data,
    weightValuePath: "Population",
  });
}

```

```

});
var population_data = [
{Asia:[
{Region: "Southern Asia", Growth: 1.32, Population: 1749046000 },
{Region: "Eastern Asia", Growth: 0.57, Population: 1620807000 },
{Region: "South-Eastern Asia", Growth: 1.20, Population: 618793000 },
{Region: "Western Asia", Growth: 1.98, Population: 245707000 },
{Region: "Central Asia", Growth: 1.43, Population: 64370000 }
] } ,
{America:[
{Region: "South America", Growth: 1.06, Population: 406740000 },
{Region: "Northern America", Growth: 0.85, Population: 355361000 },
{Region: "Central America", Growth: 1.40, Population: 167387000 },
] },
{Africa:[
{Region: "Eastern Africa", Growth: 2.89, Population: 373202000 },
{Region: "Western Africa", Growth: 2.78, Population: 331255000 },
{Region: "Northern Africa", Growth: 1.70, Population: 210002000 },
{Region: "Middle Africa", Growth: 2.79, Population: 135750000 },
{Region: "Southern Africa", Growth: 0.91, Population: 60425000 }
] }
]
];

```



[Click](#) here to view our online demo sample with Hierarchical levels.

### Layout

You can decide on the visual representation of nodes belonging to all the treemap levels using the `itemsLayoutMode` property of the TreeMap.

There are four different **TreeMap** layouts such as

- Squarified

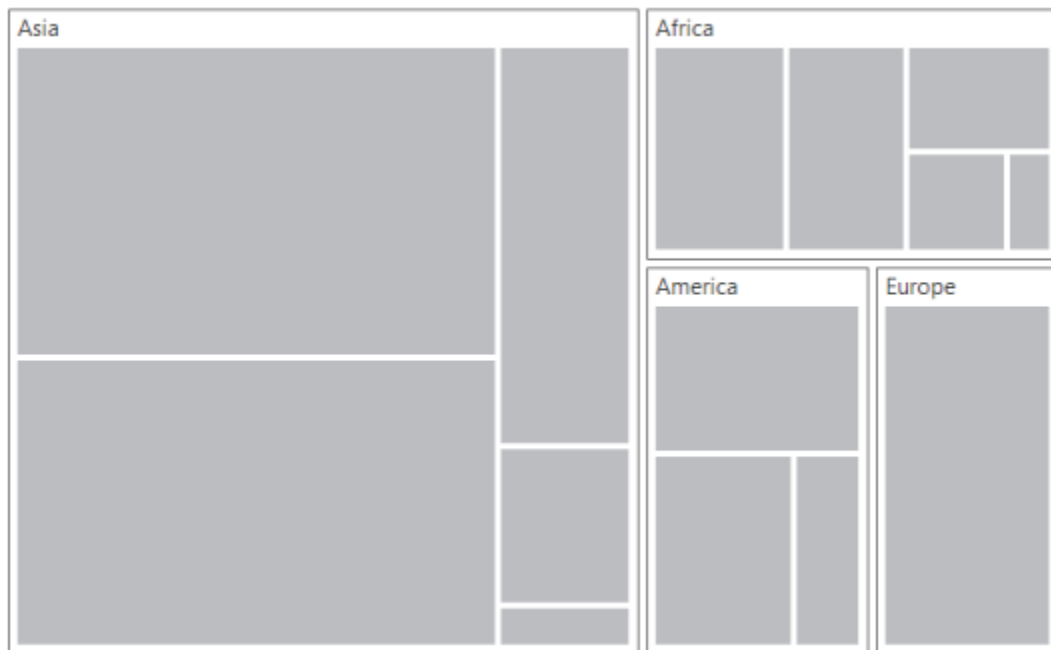
- SliceAndDiceAuto
- SliceAndDiceHorizontal
- SliceAndDiceVertical

### Squarified

**Squarified** layout creates rectangles with best aspect ratio.

#### JAVASCRIPT

```
/// <reference path="../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
module treeMapcomponent {
  $(function () {
    var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {
      dataSource: population_data,
      colorValuePath: "Growth",
      weightValuePath: "Population",
      levels: [
        { groupPath: "Continent", groupGap: 5 }
      ],
      itemsLayoutMode: "squarified"
    });
  });
}
```



Try it: [Squarified](#)

### SliceAndDiceAuto

**SliceAndDiceAuto** layout creates rectangles with high aspect ratio and displays them sorted both horizontally and vertically.

#### JAVASCRIPT

```
$(function () {  
  var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {  
    // ...  
    itemsLayoutMode:"sliceanddiceauto",  
    // ...  
  });  
});
```



Try it: [SliceAndDiceAuto](#)

[SliceAndDiceHorizontal](#)

**SliceAndDiceHorizontal** layout creates rectangles with high aspect ratio and displays them sorted horizontally.

#### JAVASCRIPT

```
$(function () {  
  var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {  
    // ...  
    itemsLayoutMode:"sliceanddicehorizontal",  
    // ...  
  });  
});
```



Try it: [SliceAndDiceHorizontal](#)

[SliceAndDiceVertical](#)

**SliceAndDiceVertical** layout creates rectangles with high aspect ratio and displays them sorted vertical.

#### JAVASCRIPT

```
$(function () {  
  var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {  
    // ...  
    itemsLayoutMode: "sliceanddicevertical",  
    // ...  
  });  
});
```





Try it: [SliceAndDiceVertical](#)

## Customization

**TreeMap** control supports color customization to determine the exact combination of colors for tree nodes displayed in **TreeMap** and tooltip support to display additional information of treemap data.

### Color

You can customize the colors of the leaf nodes of **TreeMap** using the ColorMapping support of the **TreeMap**.

ColorMapping is categorized into three different types such as,

- `uniColorMapping`
- `rangeColorMapping`
- `desaturationColorMapping`

### Uni Color Mapping

You can color, all the leaf nodes with the same color by setting the `color` value of the `uniColorMapping` property of the **TreeMap**.

### JAVASCRIPT

```

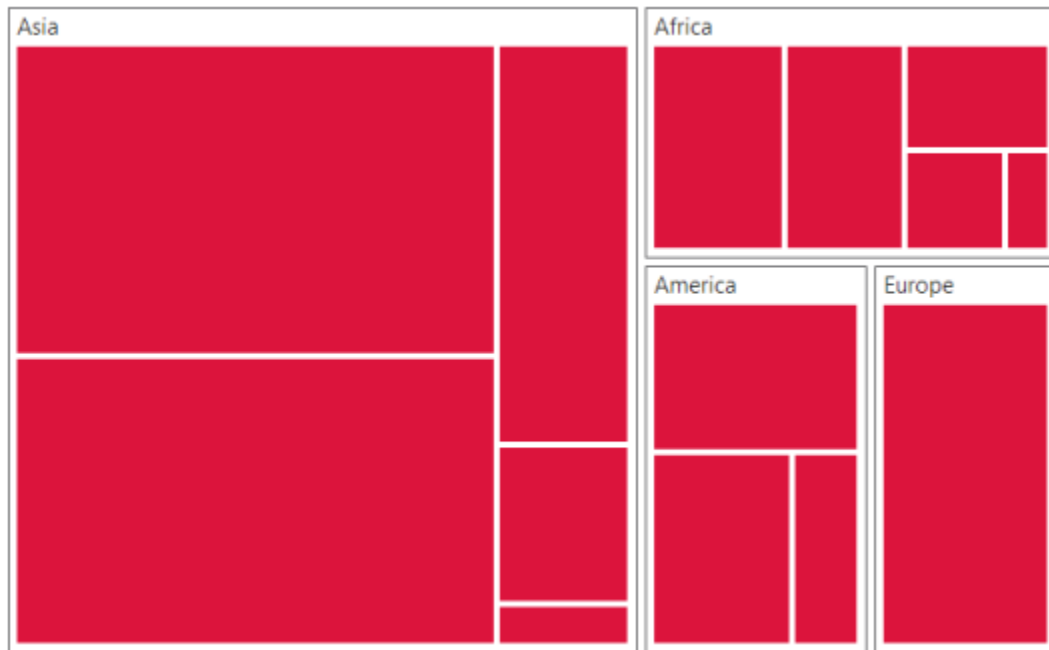
/// <reference path="../../../tsfiles/jquery.d.ts"></reference>
/// <reference path="../../../tsfiles/ej.web.all.d.ts"></reference>
module treeMapcomponent {
  $(function () {
    var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {
      // ...
      uniColorMapping: {
        color: "Crimson"
      },
      // ...
    });
  });
}

```

```

    });
  });
}

```



Try it: [UniColorMapping](#)

#### Range Color Mapping

You can group the leaf nodes based on the range of the data's color values. You can set a unique color for every ranges. To achieve this, specify the **to** and **from** values as range bound and **color** or **gradient** colors values to fill the leaf nodes of the particular range, through the **range color mapping** property of the **TreeMap**.

#### JAVASCRIPT

```

$(function () {
  var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {
    // ...
    rangeColorMapping: [
      { color: "#77D8D8", from: "0", to: "1" },
      { color: "#AED960", from: "0", to: "2" },
      { color: "#FFAF51", from: "0", to: "3" },
      { color: "#F3D240", from: "0", to: "4" }
    ],
    // ...
  });
});

```



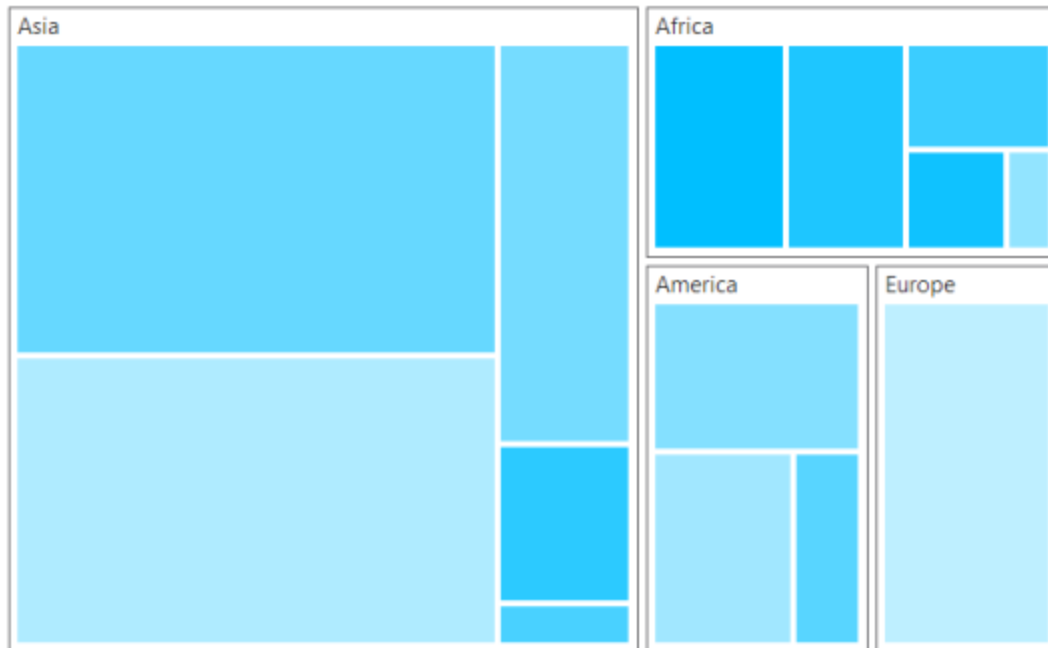
Try it: [RangeColorMapping](#)

#### *Desaturation Color Mapping*

You can differentiate all the leaf nodes using the **desaturation color mapping** property of the **TreeMap**. Differentiation is achieved, even though same color is applied for all the leaf nodes by varying the opacity of the leaf nodes based on the color value specified in the color value range using **rangeMinimum** and **rangeMaximum** value of the data collection. You can also bound the opacity range by setting **from** and **to** property of the **desaturationColorMapping**.

#### **JAVASCRIPT**

```
$(function () {  
  var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {  
    // ...  
    desaturationColorMapping: {  
      color: "DeepSkyBlue", from: "1", to: "0.2", rangeMinimum: "0",  
      rangeMaximum: "4"  
    },  
    // ...  
  });  
});
```



Try it: [DesaturationColorMapping](#)

### Tooltip

You can enable the tooltip support for the TreeMap by setting the `showTooltip` property to true. By default, it takes the property of the bound object that is referred to in the `groupPath` and displays its content when the corresponding node is tapped. The `tooltipTemplate` is a **HTML** element that is used to expose the custom template for the tooltip.

### Leaf Item Setting

You can customize the **Leaf level TreeMap items** using `leafItemSettings`. In `leafItemSettings` following customization options are available.

- You can specify the border color using `border brush` property.
- For customizing border thickness, you can use `border thickness` property.
- To customize the gap between the leaf items, you can use `gap` property.
- You can specify the label template for the leaf item using `item template` property.
- The Label and tooltip values take the property of bound object that is referred in the `labelPath` when defined.
- You can specify the position of the leaf labels using `labelPosition` property.
- You can control the mode of label visibility of the labels using `labelVisibilityMode` property.
- To show or hide the visibility of the leaf item labels you can use `showLabels` property.
- For specifying over flow action of leaf item labels you can use `textOverflow` property.

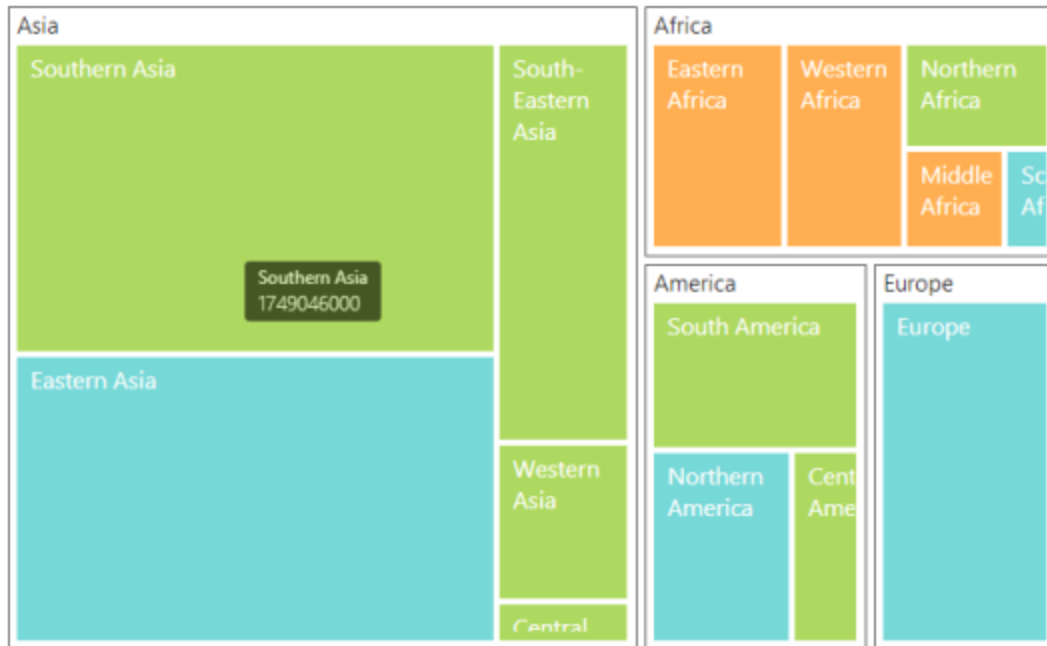
### JAVASCRIPT

```
<script type="text/javascript">
$(function () {
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {
dataSource: population_data,
colorValuePath: "Growth",
```

```

weightValuePath: "Population",
rangeColorMapping: [
  { color: "#77D8D8", from: "0", to: "1" },
  { color: "#AED960", from: "0", to: "2" },
  { color: "#FFAF51", from: "0", to: "3" },
  { color: "#F3D240", from: "0", to: "4" }
],
levels: [
  { groupPath: "Continent", groupGap: 5 }
],
leafItemSettings: { labelPath: "Region" },
showTooltip: true,
tooltipTemplate: 'template'
});
});
</script>
<script id="template" type="application/jsrender">
<div style="margin-left:17px;margin-top:-45px;">
<div style="height:auto;width:auto;background:black;border-
radius:3px;opacity:0.6">
<div style="margin-top:-20px;margin-left:9px;padding-top:3px;margin-
right:9px;">
<label style="margin-top:-20px;font-weight:normal;font-
size:12px;color:white;font-family:Segoe UI;">{{:Region}}</label>
</div>
<div style="height:10px;"></div>
<div style="margin-top:-10px;margin-left:9px;margin-right:9px;padding-
bottom:3px;">
<label style="margin-top:-10px;font-weight:normal;font-
size:14px;color:white;font-family:segoe ui
light;">{{:Population}}</label>
</div>
</div>
</div>
</script>

```



Try it: [LeafItemSettings](#)

### Border Brush

You can able to customize the border color of the treemap using the property `borderBrush`.

### JAVASCRIPT

```
//To set borderBrush API value during initialization
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"),
{borderBrush:'white'});
```

### Border Thickness

For customizing the border thickness of the treemap, you can use the `borderThickness` property.

### JAVASCRIPT

```
//To set borderThickness API value during initialization
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"),
{borderThickness:1});
```

### Dock Position

You can position the legend at top, bottom, left and right side of the treemap as per your requirement. For changing the position as per your requirement, you can use `dockPosition` property.

<ts name="ej.datavisualization.TreeMap.DockPosition"/>

Specifies the dockPosition for legend

| Name   | Description                   |
|--------|-------------------------------|
| top    | specifies the top position    |
| bottom | specifies the bottom position |

| Name  | Description                   |
|-------|-------------------------------|
| right | specifies the bottom position |
| left  | specifies the left position   |

### JAVASCRIPT

```
//To set dockPosition API value during initialization
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"),
{legendSettings:{ dockPosition: "top"}});
```

### Clicking and Dragging

You can select the single treemap element on click and drag. To click and drag treemap items, you have to enable the `draggingOnSelection` property.

### JAVASCRIPT

```
//To set draggingOnSelection API value during initialization
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"),
{draggingOnSelection:false});
```

For selecting the group element of treemap while clicking and dragging, you can use `draggingGroupOnSelection` property.

### JAVASCRIPT

```
//To set draggingGroupOnSelectionAPI value during initialization
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"),
{draggingGroupOnSelection:false});
```

### Fill with Gradient

You can customize that whether gradient color have to be applied for treemap or not. This can be customized using the property `enableGradient`.

### JAVASCRIPT

```
//To set enableGradient API value during initialization
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"),
{enableGradient:true});
```

### Responsive Treemap

You can customize whether treemap have to be responsive or not while resizing the container. For making treemap responsive you can use `enableResize` or `isResponsive` property.

### JAVASCRIPT

```
//To set enableResize API value during initialization
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"),
{enableResize:false});
```

### GroupColorMapping

You can customize the color of the each group using `groupColorMapping` property. To use group color mapping, kindly specify `groupId` and `rangeColorMapping` inside the `groupColorMapping`.

#### JAVASCRIPT

```
//To set groupColorMapping API value during initialization  
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"),  
{groupColorMapping:[{ groupId: "Asia", rangeColorMapping:[{ color:  
"#77D8D8", from: "0", to: "1"}]}]});
```

### GroupSelectionMode

You can specifies the selection mode of the treemap using `groupSelectionMode` property. You can set either group selection mode value as `Default` or `Multiple`.

#### JAVASCRIPT

```
// Set the selection mode during initialization.  
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"),  
{groupSelectionMode:'default'});
```

### Header

You can specify the header for the parent item using the property `header`. This is applicable only for hierarchical data source.

#### JAVASCRIPT

```
//To set header API value during initialization  
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"),  
{header:"Country"});
```

### Specifying HierarchicalDatasource

You can specify whether data source bound for the treemap is hierarchical or not using the property `isHierarchicalDatasource`.

#### JAVASCRIPT

```
//To set isHierarchicalDatasource API value during initialization  
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"),  
{isHierarchicalDatasource : true});
```

### Localization

You can specify the name of the culture based on which treemap is localized. To achieve this you can use the treemap property `locale`.

#### JAVASCRIPT

```
//Sets the locale value  
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {  
  locale:"en-US"  
});
```



## Treemap Items

You can specify the treemap items which you want to display in the treemap using the property `treeMapItems`.

### JAVASCRIPT

```
// Set the treeMapItems during initialization.  
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"),  
{treeMapItems:[]});
```

## TreeMap Elements

TreeMap contains various elements such as,

- Legend
- Headers
- Labels

### Legend

You can set the color value of **leaf nodes** using `treeMapLegend`. This legend is appropriate only for the **TreeMap** whose leaf nodes are colored using `rangeColorMapping`.

You can set `showLegend` property value to **"true"** to enable or disable legend visibility.

### TreeMap Legend

You can customize the treemap legend using following properties

- For customizing the alignment of legend, you can use `alignment` property.
- You can set the legend column count using `column count` property.
- To set the dock position of the legend text, you can use `dockPosition`
- You can specify the size of the legend by setting `height` and `width` of the `treeMapLegend`.
- You can decide the size of the legend icons by setting `iconWidth` and `iconHeight` properties of the `treeMapLegend` property avail in **TreeMap**.
- You can customize the left and right label text for the legend using `leftLabel` and `rightLabel` properties.
- To set the legend mode as default or interactive, you can use `mode` property.
- Using the property `template`, you can specify template for legend settings.
- To set the legend title, you can use the `title` property of the `legendSettings`.

### Label for Legend

You can customize the labels of the **legend item** using `legendLabel` property of `rangeColorMapping`.

### JAVASCRIPT

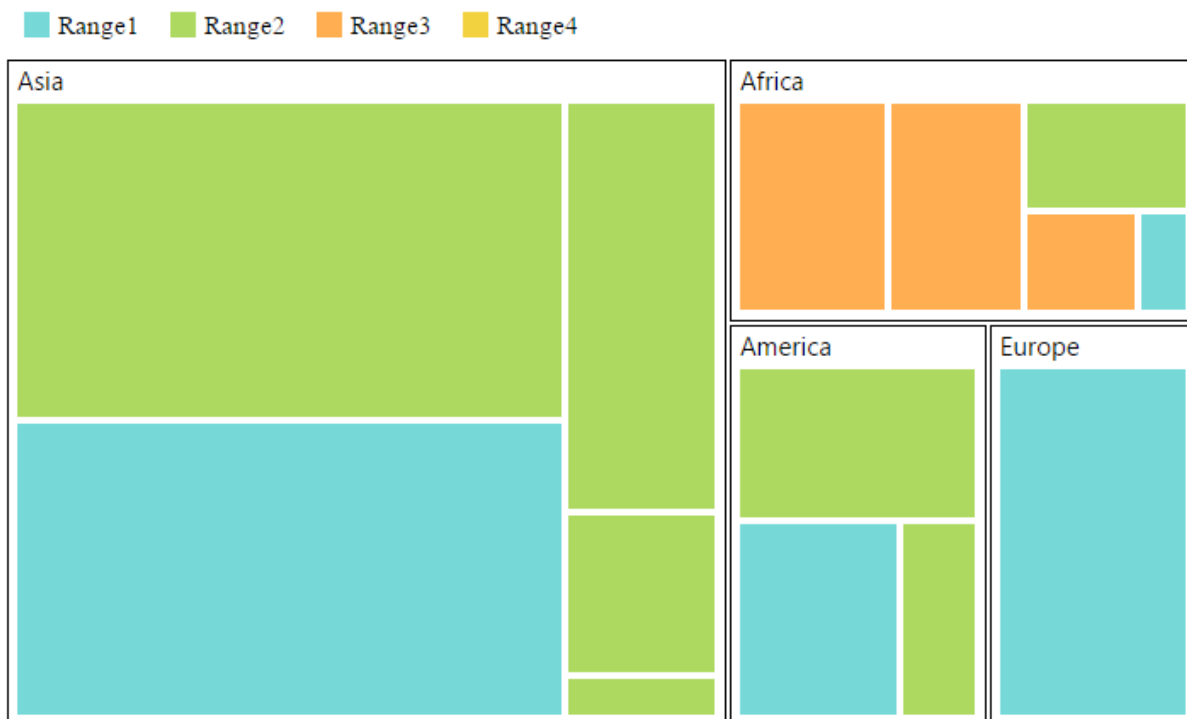
```
/// <reference path="../../../tsfiles/jquery.d.ts"></reference>  
/// <reference path="../../../tsfiles/ej.web.all.d.ts"></reference>  
module treeMapcomponent {  
  $(function () {  
    var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {  
      dataSource: population_data,  
      colorValuePath: "Growth",  

```

```

showLegend: true,
itemsLayoutMode: "Squarified",
weightValuePath: "Population",
rangeColorMapping: [
{ color: "#77D8D8", from: "0", to: "1", legendLabel: "Range1" },
{ color: "#AED960", from: "0", to: "2", legendLabel: "Range2" },
{ color: "#FFAF51", from: "0", to: "3", legendLabel: "Range3" },
{ color: "#F3D240", from: "0", to: "4", legendLabel: "Range4" }
],
levels: [
{ groupPath: "Continent", groupGap: 5 }
],
legendSettings: {
height: 40,
width: 700
}
});
});
}

```



Try it: [Legend Label](#)

#### Interactive Legend

The legends can be made interactive with an arrow mark indicating the exact range color in the legend when the mouse hovers over the corresponding treemap items. You can enable this option by setting `mode` property in `legend settings` value as "interactive" and default value of `mode` property is "default" to enable the normal legend.

#### Title for Interactive Legend

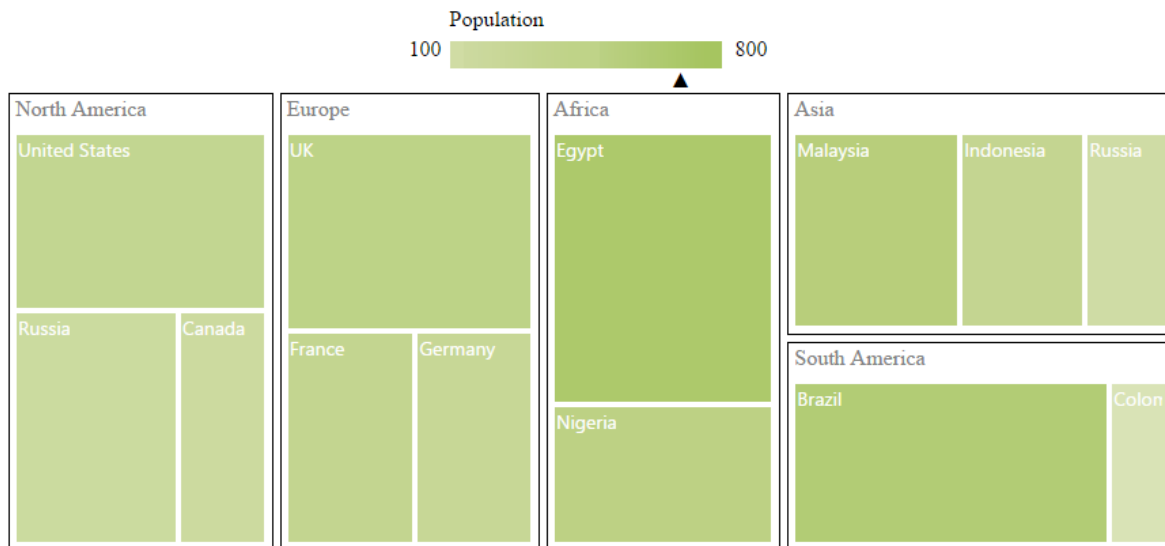
You can provide the title for interactive legend by using `title` property in `legend settings`.

### Label for Interactive Legend

You can provide the left and right labels to interactive legend by using `leftLabel` and `rightLabel` properties in `legend settings`.

### JAVASCRIPT

```
$(function () {
  var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {
    // ...
    showLegend: true,
    legendSettings: {
      height: 15,
      width: 150,
      mode: "interactive",
      title: "Population",
      leftLabel: "0.5M",
      rightLabel: "40M",
      dockPosition: "top"
    },
    // ...
  });
});
```



Try it: [Interactive Legend](#)

### Header

You can set headers for each level by setting the `showHeader` property of the each **TreeMap** levels. The `headerHeight` property helps to set the height of the header and Group path value determines the header value. You can customize the default header appearance by setting the `header template` of the **TreeMap** levels.

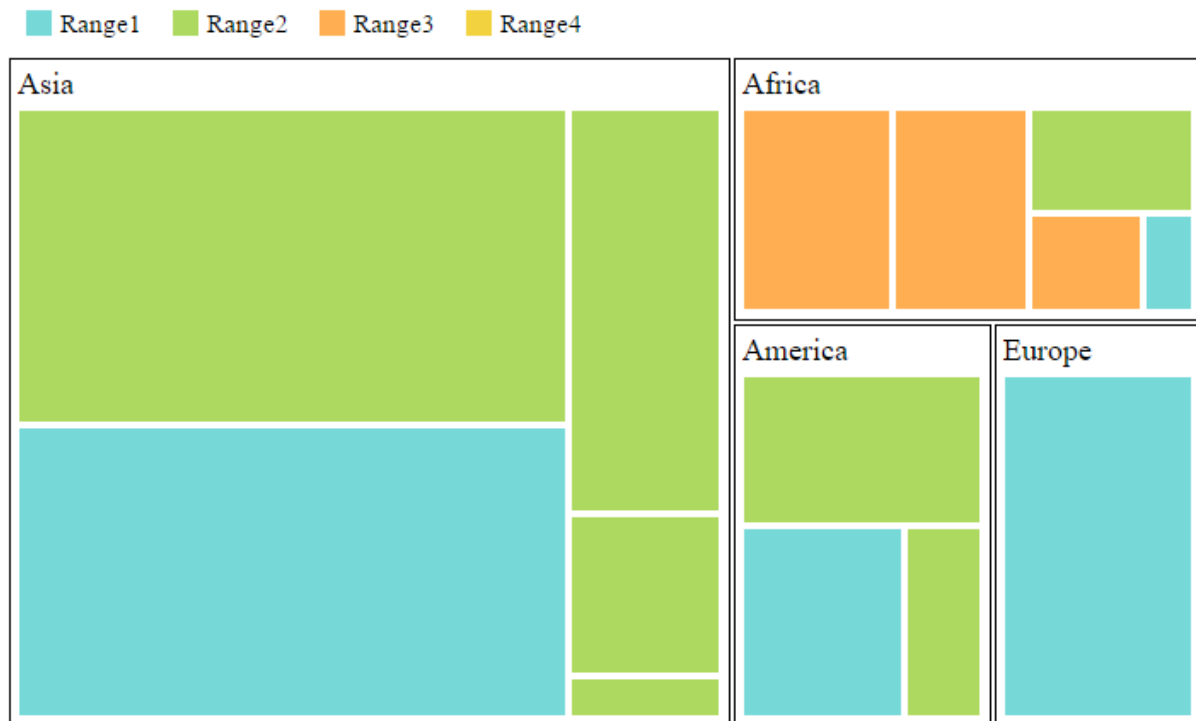
### JAVASCRIPT

```
<div id="treeMap" style="width: 950px; height: 500px; "></div>
<script type="text/javascript">
$(function () {
```

```

var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {
  // ...
  levels: [
    {groupPath: "Continent", groupGap: 2, headerHeight: 25, headerTemplate:
      'Template' }
  ],
  // ...
});
});
</script>
<script id="Template" type="application/jsrender">
<div style="background-color: white; margin:5px">
<label style="color:black;font-size:large;" >{{:header}}</label><br />
</div>
</script>

```



Try it: [Treemap Header](#)

### Customizing the header

The text in the header can be customized by triggering the event `headerTemplateRendering` of the **TreeMap**. This event is triggered before rendering the header template.

### JAVASCRIPT

```

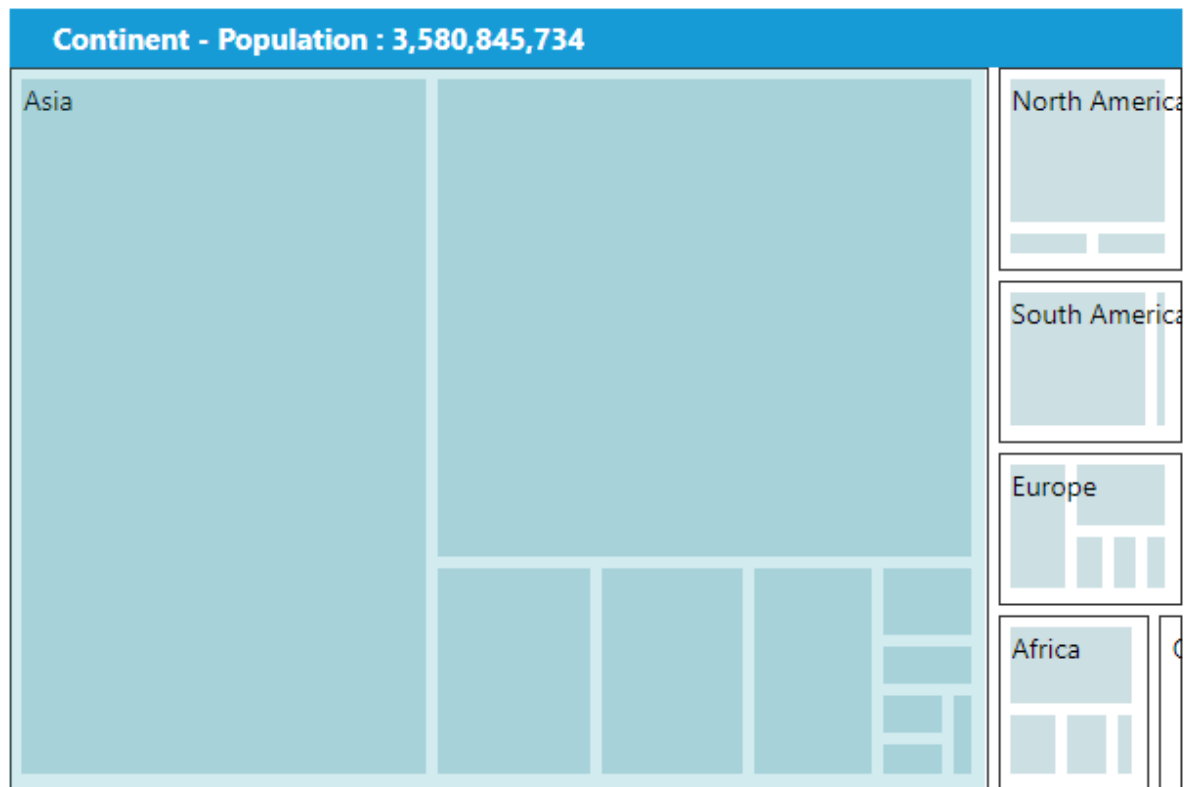
<div id="treeMap" style="width: 950px; height: 500px; "></div>
<script type="text/javascript">
$(function () {
  var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {
    // ...
    levels: [
      {groupPath: "Continent", headerHeight: 25, headerTemplate: 'Template' }
    ]
  });
});

```

```

],
headerTemplateRendering: 'loadTemplate',
// ...
});
});
</script>
<script id="Template" type="application/jsrender">
<div style="background-color: white; margin:5px">
<label style="color:black;font-size:large;" >{{:header}}</label><br />
</div>
</script>
<script>
function loadTemplate(sender) {
//...
}
</script>

```



### Label

You can also set labels for the leaf nodes by setting the `showLabels` property as true. Group path value is displayed as a label for leaf nodes. You can customize the default label appearance by setting the `labelTemplate` of the **TreeMap** levels.

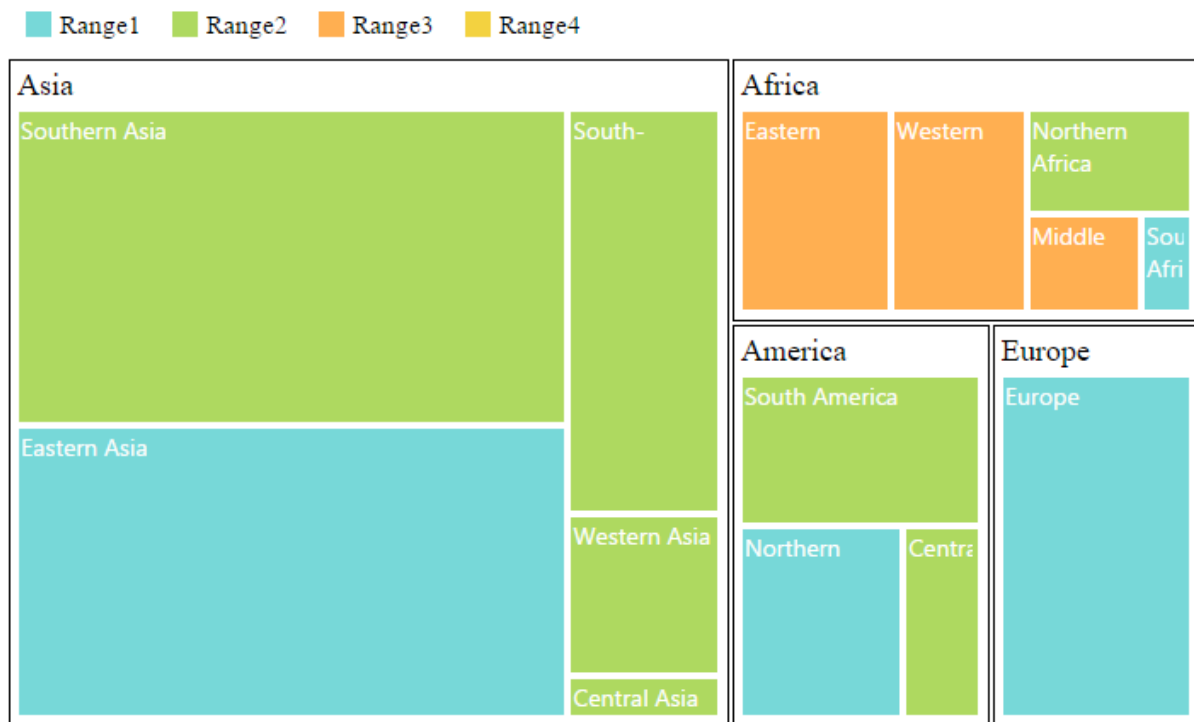
### JAVASCRIPT

```

<div id="treeMap" style="width: 1100px; height: 550px; "></div>
<script type="text/javascript">
$(function () {
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {

```

```
// ...
levels: [
{groupPath: "Continent", showLabels: true, groupGap: 2, headerHeight: 20,
headerTemplate: 'Template', labelPosition:"topLeft", }
],
leafItemSettings: { labelPath: "Region", showLabels: true},
legendSettings:{
height:40,
width:700
}
});
});
</script>
<script id="Template" type="application/jsrender">
<div style="background-color: white; margin:5px">
<label style="color:black;font-size:medium;" >{{:header}}</label><br />
</div>
</script>
```



Try it: [Treemap Label](#)

### Customizing the Overflow labels

You can handle the label overflow, by specifying any one of the following values to the property `textOverflows`

- None** - It displays the default label text.
- Hide** - You can hide the label, when it exceeds the header width.
- Wrap** - You can wrap the label text by letter.
- WrapByWord** - You can wrap the label text by word.

**JAVASCRIPT**

```

<div id="treeMap" style="width: 1100px; height: 550px; "></div>
<script type="text/javascript">
$(function () {
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {
// ...
levels: [
{groupPath: "Continent", showLabels: true, headerTemplate: 'Template' }
],
leafItemSettings: { showLabels: true , textOverFlow: "wrap"},
});
});
</script>
<script id="headertemplate" type="application/jsrender">
<div style="background-color: white; margin:5px">
<label style="color:black;font-size:medium;" >{{:header}}</label><br />
</div>
</script>

```

**Palette Color Mapping**

Treemap is having support for **palette color mapping**. You can set the color for **palette color mapping** using the property **color** in **palette color mapping**.

**JAVASCRIPT**

```

//To set the colors of the paletteColorMapping during initialization
var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"),
{paletteColorMapping{colors: ["red","green","blue", "yellow"]}});

```

**Drill Down Support**

**Treemap** enables drill down to expose the hierarchy achieved by clicking on a node and this results in enabling the **Treemap** to move to the next level or sub level and can return back to the normal **Treemap** view by clicking on the node header. Only a single level of the **Treemap** is visible at once.

**Enable Drill Down**

**Treemap** elements can be drilled down by setting the **enableDrillDown** property to true. You can view the hierarchy of the **Treemap** by clicking on the treemap items and can move to the previous level by clicking on the drill down header. The header color can be customized by changing the values in the property **drillDownHeaderColor** and the selection color can be done by changing the **drillDownSelectionColor** property.

| Property                | Type   | Description  |
|-------------------------|--------|--|
| enableDrillDown         | bool   | Gets or sets a value that indicates whether the drill down feature is enabled or not |
| drillDownHeaderColor    | string | Gets or sets a color for header during drill down                                    |
| drillDownSelectionColor | string | Gets or sets a color for highlighting tree map item during drill down.               |

**JAVASCRIPT**

```

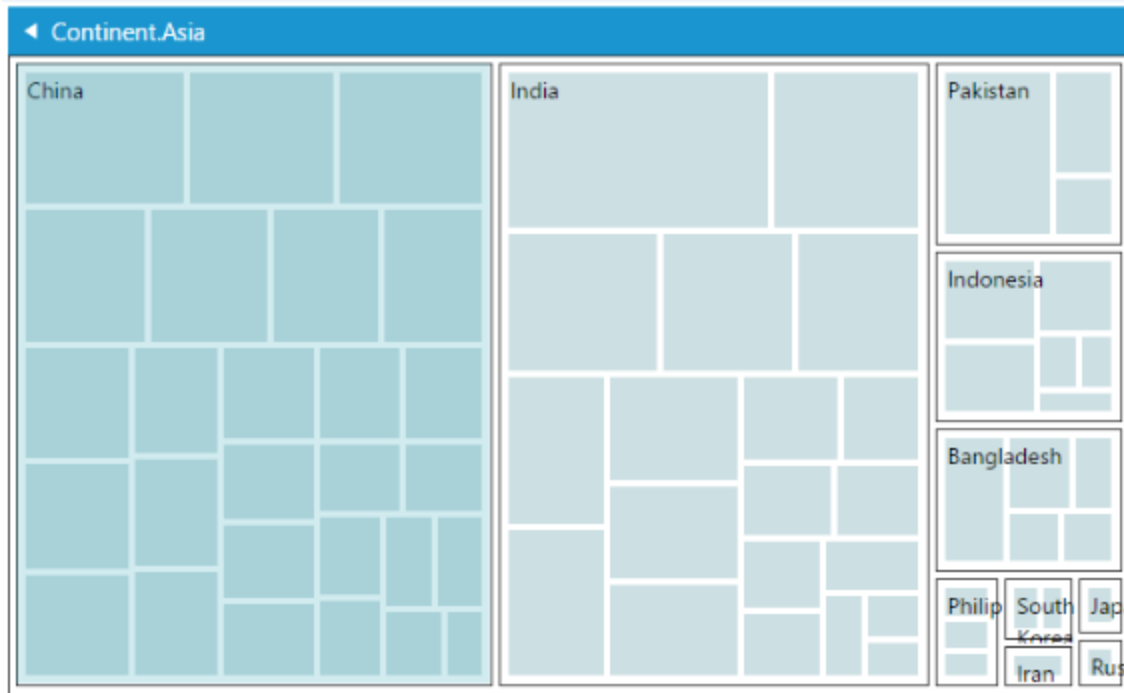
<div id="treeMap" style="width: 700px;height:400px;"></div>
<script type="text/javascript">
  /// <reference path="../../tsfiles/jquery.d.ts"></reference>
  /// <reference path="../../tsfiles/ej.web.all.d.ts"></reference>
  module treeMapcomponent {
    $(function () {
      var treeMapSample = new ej.datavisualization.TreeMap($("#treeMap"), {
        dataSource: population_data,
        enableDrillDown: true,
        drillDownHeaderColor: "#199DAF",
        drillDownSelectionColor: "#199DAF",
        uniColorMapping: { color: "#CCDFE3" },
        weightValuePath: "Population",
        levels: [
          { groupPath: "Continent", showLabels: true, groupGap: 5, headerHeight:
            25, showHeader: true, headerTemplate: 'Template' },
          { groupPath: "Country", showLabels: true, groupGap: 0, headerHeight: 25,
            showHeader: true, headerTemplate: 'Template' },
          { groupPath: "Name", showLabels: true, groupGap: 0, headerHeight: 25,
            showHeader: true, headerTemplate: 'Template' }
        ]
      });
    });
  }
</script>

```



*Before Drill Down*





*After Drill Down*

Try it: [DrillDown](#)

## AngularJS Support

**AngularJS** is a **JavaScript** Framework added to a **HTML** page with a **<script>** tag. It extends **HTML** attributes with directives and binds data to **HTML** with expressions. **AngularJS** directives allow you to specify custom and reusable **HTML** tags that moderate the behavior of certain elements. **Angular binding** uses directives to plug its action into the page. Directives, all prefaced with **ng-**, are placed in **HTML** attributes. To know more about **Angular binding** refer

to: <https://help.syncfusion.com/js/angularjs>

Apply the plugin and property assigning the **Treemap** element through the directive that starts with the letter **"e-"**. The following code illustrates how to bind data to the **Treemap** component through **Angularsupport**.

## JAVASCRIPT

```
var population_data = [
  { Continent: "Asia", Country: "Indonesia", Growth: 3, Population:
    237641326 },
  { Continent: "Asia", Country: "Russia", Growth: 2, Population: 152518015
  },
  { Continent: "Asia", Country: "Malaysia", Growth: 1, Population: 29672000
  },
  { Continent: "North America", Country: "United States", Growth: 4,
    Population: 315645000 },
  { Continent: "North America", Country: "Mexico", Growth: 2, Population:
    112336538 },
  { Continent: "North America", Country: "Canada", Growth: 1, Population:
    39056064 },
```

```
{ Continent: "South America", Country: "Colombia", Growth: 1, Population: 47000000 },
{ Continent: "South America", Country: "Brazil", Growth: 3, Population: 193946886 },
{ Continent: "Africa", Country: "Nigeria", Growth: 2, Population: 170901000 },
{ Continent: "Africa", Country: "Egypt", Growth: 1, Population: 83661000 },
{ Continent: "Europe", Country: "Germany", Growth: 1, Population: 81993000 },
{ Continent: "Europe", Country: "France", Growth: 1, Population: 65605000 },
{ Continent: "Europe", Country: "UK", Growth: 1, Population: 63181775 },
];
```

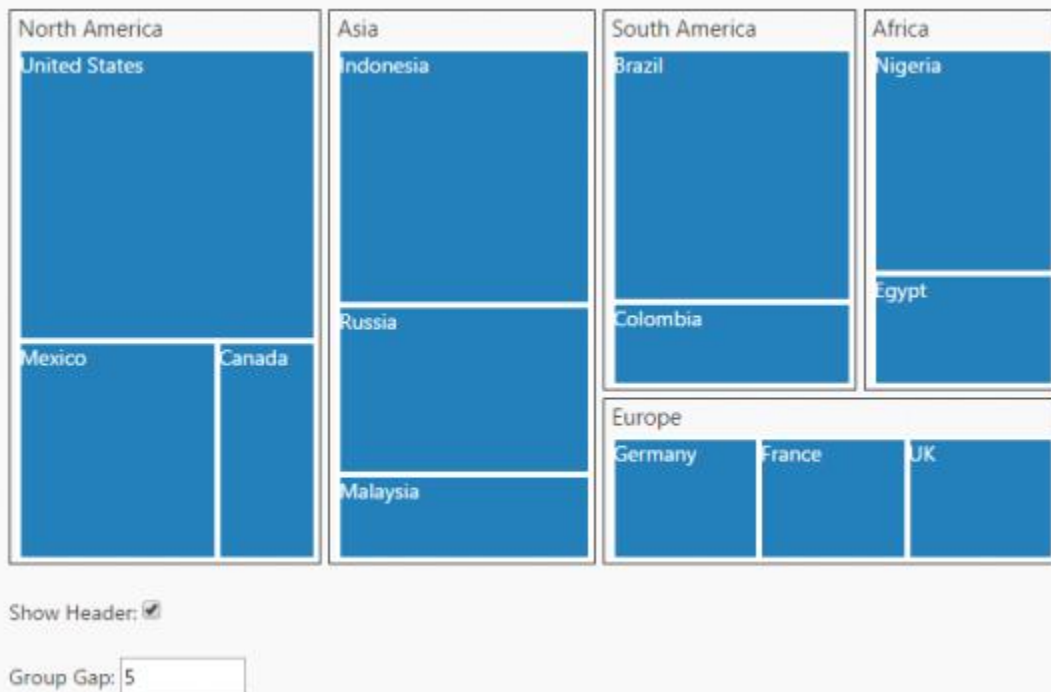
## JAVASCRIPT

```
//References to be added for AngularJS support.
<script src="
https://ajax.googleapis.com/ajax/libs/angularjs/1.0.1/angular.min.js"></script>
<script src=" http://cdn.syncfusion.com/js/ej.widget.angular-
latest.min.js"></script>
<div ng-app="SyncApp">
//Initializes controller
<div ng-controller="TreeMapController">
//Initializes TreeMap
<div id="treemap" ej-treemap e-datasource="datasource" e-unicolormapping-
color="color" e-weightvaluepath="weightValuePath" e-
colorvaluepath="colorValuePath" e-leafitemsettings-labelpath="labelPath"
style="width: 700px;height:370px;">
<e-levels>
<e-level e-grouppath="groupPath" e-groupgap="groupGap"
e-showheader="showHeader">
</e-level>
</e-levels>
</div>
//Renders a checkbox to change the header visibility
<div>
Show Header: <input type="checkbox" ng-model="showHeader"
style="outline: none;"/>
</div>
//Renders a textbox to change the groupGap value
<div>
Group Gap: <input type="text" id="Text11" ng-model="groupGap"
style="width: 110px" />
</div>
<script>
angular.module('syncApp', ['ejangular'])
.controller('TreeMapController', function ($scope) {
$scope.datasource = population_data;
$scope.colorValuePath = "Growth";
$scope.weightValuePath = "Growth";
$scope.labelPath = "Country";
$scope.groupPath = "Continent";
$scope.groupGap = 5;
```

```

$scope.showHeader = true;
$scope.color = "#2380BB";
});
</script>
</div>
</div>

```



Try it: [Treemap AngularJS Support](#)

Methods

*refresh()*

Method to reload treemap with updated values.

### HTML

```
<div id="treeMap"></div>
```

### JAVASCRIPT

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module LinearGaugeComponent {
$(function () {
var sample = new ej.datavisualization.TreeMap($("#treeMap"), {
///...///
});
sample.refresh();
});
});
}

```

## Events

### *treeMapItemSelected*

Triggers on treemap item selected.

#### **JAVASCRIPT**

```
<script>
//treeMapItemSelected event for tree map
$(function () {
  var sample = new ej.datavisualization.TreeMap($("#treeMap"), {
    treeMapItemSelected: function () {
      //...//
    }
  });
});
</script>
```

### *drillStarted*

Triggers when drilldown is started

#### **JAVASCRIPT**

```
<script>
//drillStarted event for tree map
$(function () {
  var sample = new ej.datavisualization.TreeMap($("#treeMap"), {
    drillStarted: function () {
      //...//
    }
  });
});
</script>
```

### *drillDownItemSelected*

Triggers on treemap drilldown item selected.

#### **JAVASCRIPT**

```
<script>
//drillDownItemSelected event for tree map
$(function () {
  var sample = new ej.datavisualization.TreeMap($("#treeMap"), {
    drillDownItemSelected: function () {
      //...//
    }
  });
});
</script>
```

### *headerTemplateRendering*

Triggers before rendering the treemap drilldown header template

#### **JAVASCRIPT**

```
<script>
//headerTemplateRendering event for tree map
$(function () {
var sample = new ej.datavisualization.TreeMap($("#treeMap"), {
headerTemplateRendering: function () {
//...//
}
});
});
</script>
```

### *refreshed*

Triggers after refreshing the treemap items.

#### **JAVASCRIPT**

```
<script>
//refreshed event for tree map
$(function () {
var sample = new ej.datavisualization.TreeMap($("#treeMap"), {
refreshed: function () {
//...//
}
});
});
</script>
```

### *treeMapGroupSelected*

Triggers when the group selection is performed on treemap items.

#### **JAVASCRIPT**

```
<script>
//treeMapGroupSelected event for tree map
$(function () {
var sample = new ej.datavisualization.TreeMap($("#treeMap"), {
treeMapGroupSelected: function () {
//...//
}
});
});
</script>
```

## TreeView

### Overview

TreeView control for JavaScript represents hierarchical data in tree like structure with advanced functions to perform edit, drag and drop, selection with checkboxes and more. TreeView can be generated from 'UL LI' elements, JSON data source or using OData services.

### Key features

- [Data population](#)

- [Node manipulations](#)
- [Drag and drop](#)
- [Template](#)
- [Checkbox](#)
- Accessibility
- Appearance and styling

## Getting Started

Using the following steps, you can create a **TypeScript** TreeView component.

### Creating an TreeView in TypeScript

You can create a **TypeScript** application with the help of the given [TypeScript Getting Started Documentation](#).

Within an index.html file and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<title>TypeScript Application</title>
<link
href="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/fl
at-azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script
src="http://cdn.syncfusion.com/**{{**site.releaseversion**}}**/js/web/ej.
web.all.min.js" type="text/javascript"></script>
</head>
<body>
<!--Add TreeView sample here-->
</body>
</html>
```

Execute the below steps to render the TreeView control in TypeScript.

#### HTML

```
<div style="width: 250px; max-width:100%">
<ul id="treeView">
<li class="expanded">
Artwork
<ul>
<li>
Abstract
<ul>
<li>2 Acrylic Mediums</li>
<li>Creative Acrylic</li>
<li>Modern Painting</li>
<li>Canvas Art</li>
<li>Black white</li>
</ul>
</li>
```

```
<li>
Children
<ul>
<li>Preschool Crafts</li>
<li>School-age Crafts</li>
<li>Fabulous Toddler</li>
</ul>
</li>
<li>
Comic / Cartoon
<ul>
<li>Batman</li>
<li>Adventures of Superman</li>
<li>Super boy</li>
</ul>
</li>
</ul>
</li>
<li class="expanded">
Books
<ul>
<li>
Comics
<ul>
<li>The Flash</li>
<li>Human Target</li>
<li>Birds of Prey</li>
</ul>
</li>
<li>Entertaining</li>
<li>Design</li>
</ul>
</li>
<li>
Music
<ul>
<li>
Classical
<ul>
<li>Medieval</li>
<li>Orchestral</li>
</ul>
</li>
<li>Mass</li>
<li>Folk</li>
</ul>
</li>
</ul>
</div>
<script src="app.js"></script>
```

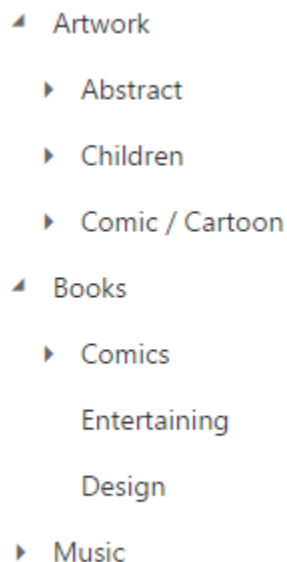
- Create app.ts file and use the below content

## JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module TreeViewComponent {
  $(function () {
    var tree = new ej.TreeView($("#treeView"), {
    });
  });
}
```

- Now build your application, so that the **app.ts** file will be compiled and automatically generate the **app.js** file which is added to your project (User has nothing to do with this file). Now, whatever code changes that you make in **app.ts** file will be reflected in **app.js** file by compiling and building the application.

Execution of above code will render the following output.



### Drag and Drop

You can drag and drop tree nodes between two TreeView by setting `allowDragAndDrop` as `true` along with `allowDragAndDropAcrossControl` as `true`.

### JS

```
/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module TreeViewComponent {
  $(function () {
    var tree = new ej.TreeView($("#treeView"), {
      allowEditing: true,
      allowDragAndDrop: true,
      allowDropChild: true,
      allowDropSibling: true,
    });
  });
}
```



```
}

```

### Checkboxes

TreeView consists of in-built checkbox option and it can be displayed to the left of the tree node by setting the showCheckbox property as true. It allows you to select more than one node at a time.

### JS

```

/// <reference path="jquery.d.ts" />
/// <reference path="ej.web.all.d.ts" />
module TreeViewComponent {
  $(function () {
    var tree = new ej.TreeView($("#treeView"), {
      showCheckbox : true,
      autoCheckParentNode : true
    });
  });
}

```

### Populate Data

TreeView can be populated with local or remote data source by using a property [dataSource](#), which is the member of [fields](#) property.

In TreeView, you should use “**fields**” property to go with data source. It specifies the mapping fields for the data source to receive the data, query to process the data and field mappers to map the data members.

### Fields

The following table contains the list of members with description for **fields** property.

| Properties                 | Description   |
|----------------------------|---|
| <a href="#">dataSource</a> | The data source contains the list of data for generating the TreeView list.   |
| <a href="#">query</a>      | It specifies the query to retrieve the data from the online server.   |
| <a href="#">tableName</a>  | It specifies the name of the table from which data to be processed from given data source.                                      |
| <a href="#">id</a>         | It specifies the ID of the node.  |
| <a href="#">parentId</a>   | It specifies the parent id of the node  |
| <a href="#">text</a>       | It specifies the text content of the node.  |
| <a href="#">hasChild</a>   | It specifies the node has child (which is the nested or inner level of nodes). Also itâ€™s used in load on demand of tree data. |

|                                |   |
|--------------------------------|---|
| <a href="#">expanded</a>       | It specifies the tree node to be in expanded state  |
| <a href="#">selected</a>       | It specifies the select node at initialize. N> only one node get selected by default. If you enable multiple selection in TreeView then you can able to select one or more nodes at initialize. |
| <a href="#">isChecked</a>      | It specifies the node to be in checked state, if tree node represented with checkboxes.   |
| <a href="#">imageUrl</a>       | It defines the image location.  |
| <a href="#">imageAttribute</a> | It defines the image attributes such as height, width, styles, etc.   |
| <a href="#">spriteCssClass</a> | It defines the sprite CSS for the image tag.  |
| <a href="#">htmlAttribute</a>  | It defines the HTML attributes such as class and styles for a node ("li" tag).  |
| <a href="#">linkAttribute</a>  | It defines the HTML attributes such as class and styles for a link tag, which is child of node.   |

Mapping all fields members with corresponding key from the array of JSON data.

### JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
  var localData = [
    { id: 1, text: "Item 1", expanded: true, nodeProperty: { class: "textBlue", value: "Item 1" } },
    { id: 2, text: "Item 2", linkProperty: { class: "textUnderline", href: "http://www.syncfusion.com", target: "_blank" } },
    { id: 3, text: "Item 3", selected: true, spriteImage: "mail-icon sprite-calendar" },
    { id: 4, text: "Item 4", checked: true, imageProperty: { width: 20, height: 20 }, imageUrl: "http://cdn.syncfusion.com/13.3.0.7/js/web/flat-azure/images/ajax-loader.gif" },
    { id: 5, parent: 1, text: "Item 1.1" },
    { id: 6, parent: 1, text: "Item 1.2" },
    { id: 7, parent: 1, text: "Item 1.3" },
    { id: 8, parent: 3, text: "Item 3.1" },
    { id: 9, parent: 3, text: "Item 3.2" },
    { id: 10, parent: 5, text: "Item 1.1.1" }
  ];
  $(function () {
    var tree = new ej.TreeView($("#treeView"), {
      showCheckbox: true,
      fields: {

```

```

id: "id",
text: "text",
parentId: "parent",
dataSource: localData,
isChecked: "checked",
selected: "selected",
spriteCssClass: "spriteImage",
imageUrl: "imageUrl",
htmlAttribute: "nodeProperty",
linkAttribute: "linkProperty",
imageAttribute: "imageProperty"
}
});
});
}

```

**Note:** If you want to display nodes in Root level, exclude **parentId** attribute or specify **"0"** in corresponding value.

To find the complete details about fields, refer the sample [here](#).

### Local Data

TreeView can be rendered from a self-referential data by providing the two required fields [id](#) and [parentId](#).

### JS

```

var localData = [
{ id: 1, text: "Item 1" },
{ id: 2, text: "Item 2" },
{ id: 3, text: "Item 3" },
{ id: 4, text: "Item 4" },
{ id: 5, parent: 1, text: "Item 1.1" }
];

```

Above flat array of JSON data can be directly assigned to [dataSource](#) property and mapping data fields with respect to the mapper field in order to create TreeView.

### HTML

```

<!--create the TreeView wrapper-->
<div id="treeView">
</div>

```

### JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
var localData = [
{ id: 1, text: "Item 1" },
{ id: 2, text: "Item 2" },
{ id: 3, text: "Item 3" },
{ id: 4, text: "Item 4" },
{ id: 5, parent: 1, text: "Item 1.1" },

```

```

{ id: 6, parent: 1, text: "Item 1.2" },
{ id: 7, parent: 1, text: "Item 1.3" },
{ id: 8, parent: 3, text: "Item 3.1" },
{ id: 9, parent: 3, text: "Item 3.2" },
{ id: 10, parent: 5, text: "Item 1.1.1" }
];
$(function () {
    // initialize and bind the TreeView with local data
    var tree = new ej.TreeView($("#treeView"), {
        fields: { dataSource: localData, id: "id", parentId: "parent", text:
            "text" }
    });
});

```

### Nested Object Support

The nested object support is provided for the TreeView component. Please find the following JSON.

#### JS

```

var localData = [
    { id: 1, name: { nodeName: "Discover Music"}, hasChild: true, expanded:
        true },
    { id: 2, parent: 1, name: {nodeName:"Hot Singles" }},
    { id: 3, parent: 1, name: {nodeName:"Rising Artists" }},
    { id: 4, parent: 1, name:{nodeName: "Live Music" }}
];

```

Above flat array of JSON data can be directly assigned to [dataSource](#) property and mapping data fields with respect to the mapper field in order to create TreeView.

#### HTML

```

<!--create the TreeView wrapper-->
<div id="treeView">
</div>

```

#### JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
    var localData = [
        { id: 1, name: { nodeName: "Discover Music"}, hasChild: true, expanded:
            true },
        { id: 2, parent: 1, name: {nodeName:"Hot Singles" }},
        { id: 3, parent: 1, name: {nodeName:"Rising Artists" }},
        { id: 4, parent: 1, name:{nodeName: "Live Music" }}
    ];
    $(function () {
        // initialize and bind the TreeView with local data contains nested
        object
        var tree = new ej.TreeView($("#treeView"), {
            fields: { dataSource: localData, id: "id", parentId: "parent", text:
                "name.nodeName" }
        });
    });
}

```

```
});  
});  
}
```

### Remote Data

When using remote data binding, the adaptor of [ej.DataManager](#) plays vital role in processing queries to make them suitable to send along with data request and also process the response data from the server.

### OData

**OData** is a standardized protocol for creating and consuming data. You can bind [oData service](#) data to TreeView in two ways using DataManager.

- Passing OData service link to DataManager

You can provide the OData service URL directly to the [ej.DataManager](#) class and then we can assign it to TreeView dataSource.

### HTML

```
<!--create the TreeView wrapper-->  
<div id="treeView">  
</div>
```

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module TreeViewComponent {  
    var parentData =  
        ej.DataManager("http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Categories"),  
        childData =  
        ej.DataManager("http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Products");  
    $(function () {  
        // initialize and bind the TreeView with remote data  
        var tree = new ej.TreeView($("#treeView"), {  
            fields: {  
                dataSource: parentData,  
                id: "CategoryID", text: "CategoryName",  
                child: {  
                    dataSource: childData,  
                    id: "ProductID", parentId: "CategoryID", text: "ProductName"  
                }  
            }  
        });  
    });  
}
```

### Load on Demand

Load on demand is a technique (Lazy load) that is used to reduce the bandwidth size of consuming huge data. You can load data on demand in TreeView by using [loadOnDemand](#) property when you're going to use huge data.

For local data source, TreeView loads the first level nodes initially. While expand a parent node then TreeView loads it's their child nodes from the data source based on the parentId member. It reduces the time to render TreeView with huge data.

Refer below code example to load data on demand from local data source.

### HTML

```
<!--create the TreeView wrapper-->
<div id="treeview"></div>
```

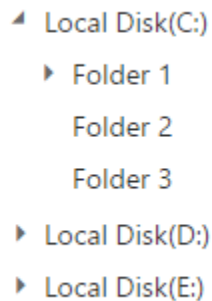
### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
  var localData = [
    { id: 1, name: "Local Disk(C:)", hasChild: true },
    { id: 2, name: "Local Disk(D:)", hasChild: true },
    { id: 3, name: "Local Disk(E:)", hasChild: true },
    { id: 4, parentId: 1, name: "Folder 1", hasChild: true },
    { id: 5, parentId: 1, name: "Folder 2" },
    { id: 6, parentId: 1, name: "Folder 3" },
    { id: 7, parentId: 2, name: "Folder 4" },
    { id: 8, parentId: 2, name: "Folder 5", hasChild: true },
    { id: 9, parentId: 2, name: "Folder 6" },
    { id: 10, parentId: 3, name: "Folder 7" },
    { id: 11, parentId: 3, name: "Folder 8" },
    { id: 12, parentId: 3, name: "Folder 9", hasChild: true },
    { id: 13, parentId: 4, name: "File 1" },
    { id: 14, parentId: 4, name: "File 2" },
    { id: 15, parentId: 4, name: "File 3" },
    { id: 16, parentId: 8, name: "File 4" },
    { id: 17, parentId: 8, name: "File 5" },
    { id: 18, parentId: 8, name: "File 6" },
    { id: 19, parentId: 12, name: "File 7" },
    { id: 20, parentId: 12, name: "File 8" },
    { id: 21, parentId: 12, name: "File 9" }
  ];
  $(function () {
    var tree = new ej.TreeView($("#treeView"), {
      loadOnDemand: true,
      fields: { dataSource: localData, id: "id", parentId: "parentId", text:
        "name", hasChild: "hasChild" },
    });
  });
}
```

The following screenshot displays the load on demand for local data source in TreeView control.



While expanding the parent node



After expanding the parent node

For more details about load on demand for local data source, refer the sample [here](#).

For remote data source, TreeView loads the first level nodes initially. While expand the node from TreeView, the data manager passes the query to the controller. Based on this query, you can filter the data from table and return to TreeView.

Refer below code example to load data on demand from remote data source.

### HTML

```
<!--create the TreeView wrapper-->
<div id="treeview"></div>
```

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
$(function () {
// DataManager creation
var dataManger = ej.DataManager({
url: "http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/"
});
// Query creation
var query =
ej.Query().from("Categories").select("CategoryID,CategoryName").take(3);
var tree = new ej.TreeView($("#treeview"), {
loadOnDemand: true,
fields: {
dataSource: dataManger, query: query, id: "CategoryID", text:
"CategoryName", hasChild: "CategoryName",
child: { dataSource: dataManger, tableName: "Products", parentId:
"CategoryID", text: "ProductName" }
}
```

```

}
});
});
}

```

## Tree Node

TreeView node is structured with expand/collapse arrow, checkbox, image and text as shown in below.



Also TreeView node object holds the following properties.

| Properties | Data Type | Description                |
|------------|-----------|----------------------------|
| checked    | Boolean   | Checked state of the node  |
| count      | Number    | Number of child            |
| expanded   | Boolean   | Expanded state of the node |
| index      | Number    | Index of the node          |
| level      | Number    | Level of the node          |
| id         | String    | Node id                    |
| parentId   | String    | Parent id of the node      |
| selected   | Boolean   | Selected state of the node |

## Get/Set Node Value

TreeView provides a set of options to configure all its properties by setting and getting values at initialization or dynamically.

To get the node value, you can use [getNode](#) method as shown in the below code example, in which on button click action the node value has retrieved.

Also you can get the text value of tree node by using [getText](#) method.

## JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
    $(function () {
        var tree = new ej.TreeView($("#treeView"), {
            fields: {
                dataSource: [
                    {
                        id: 1, text: "Item 1",
                        child: [
                            { id: 4, text: "Item 1.1" },

```



```
{ id: 5, text: "Item 1.2" }  
],  
},  
{ id: 2, text: "Item 2" },  
{ id: 3, text: "Item 3" }  
]  
}  
});  
});  
}  
  
//bind below onClick action to button to get node at button click or else  
in any action  
function onClick() {  
var node = tree.getNode("1");  
}
```

**Note:** Existing TreeView instance can be created by [jQuery.data\(\)](#) and you can control the API's of TreeView behavior.

To edit the node text, you can use [updateText](#) method as shown below code example.

#### JS

```
//get node object using getNode method  
var node = tree.getNode("2");  
//sets text to existing node  
tree.updateText(node.id, "updated Item");
```

#### Get Parent Node

To get current parent node of a particular node, you can use the [getParent](#) method as shown in below code example

#### JS

```
//create an instance from an existing TreeView.  
// only after control creation we can get treeObj otherwise it throws  
exception.  
var node = tree.getParent("4");
```

#### Get Node Index

To get node index you can use the [getNodeIndex](#) as shown in below code example

You can use [getNodeByIndex](#) method to get TreeView node by using index position.

#### JS

```
//create an instance from an existing TreeView.  
var node = tree.getNode("4");  
//gets the index of node  
tree.getNodeIndex(node.id);
```

#### Node Manipulations

You can perform following operation in tree nodes and the modified node values can be saved in database.

### Add or Remove nodes

To add/remove nodes programmatically, use [addNode](#) and [removeNode](#) methods of the TreeView.

#### JS

```
//create an instance from an existing TreeView.
tree.addNode(newNode, "2");
//to remove node
tree.removeNode("4");
```

You can able to add a new node after or before specific TreeView node by using [insertAfter](#) and [insertBefore](#) methods.

#### JS

```
//create an instance from an existing TreeView.
var newNode = { id: 12, text: "Item 2.2" };
//to add tree node after some element, which having id 2
tree.insertAfter(newNode, "2");
var newNode = { id: 13, text: "Item 2.3" };
//to add tree node before some element, which having id 2
tree.insertBefore(newNode, "2");
```

### Move node

You can also achieve cut and paste operation by using [moveNode](#) methods.

#### JS

```
//create an instance from an existing TreeView.
//to move tree node which having id 5, to child of node which having id 2.
tree.moveNode("5", "2");
```

### Expand or Collapse node

Tree nodes can be expanded or collapsed by clicking the expand/collapse icon of the node or in code behind by using the following methods.

| Methods                      | Description                         |
|------------------------------|-------------------------------------|
| <a href="#">expandNode</a>   | Expands the node with specified id  |
| <a href="#">collapseNode</a> | Collapse the node with specified id |
| <a href="#">expandAll</a>    | Expands all the node                |
| <a href="#">collapseAll</a>  | Collapse all the node               |

Also you can get all the expanded nodes index in tree by using [getExpandedNodesIndex](#) method, which returns the array of expanded node indices.

### Prevent multiple node expand

You can able to prevent multiple expand of TreeView nodes by specifying [enableMultipleExpand](#) as false.

For example, if you want to allow only one node to be expanded in TreeView at a time. Refer the sample [here](#).

### JS

```
// Initialize and bind the TreeView with enableMultipleExpand property
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
  $(function () {
    var tree = new ej.TreeView($("#treeView"), {
      enableMultipleExpand: false,
      dataSource: localData,
      fields: {
        id: "id",
        text: "text",
        parentId: "parent"
      }
    });
  });
}
```

### Get updated node collection

You can get the updated node values after manipulating or editing the node of TreeView to save at server end using [getTreeData](#) method. It returns the JSON data represented as in tree with modified structure.

You can also get the updated data source for remote data binding after performing the operation like editing, selecting/unselecting, expanding/collapsing, checking/unchecking and removing node. You cannot get the updated data source when you perform operation like drag and drop, adding node for remote data binding.

The updated data source also contains custom attributes if you return these from server.

### JS

```
//create an instance from an existing TreeView.
//to get TreeView data
tree.getTreeData();
```

### Editing

You can directly edit the tree node's text in-place by double-click on the tree node or select the tree node and press F2 key. The editing works only if the [allowEditing](#) property is true in TreeView control. When editing is completed by focus out or "enter" key press, the modified node's text is saved automatically. The [nodeEdit](#) event will be triggered whenever edited the TreeView node.

Also [beforeEdit](#) event will be triggered before the TreeView node change into editing mode.

### JS

```
// Initialize and bind the TreeView with allowEditing property
```

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
  $(function () {
    var tree = new ej.TreeView($("#treeView"), {
      allowEditing: true,
      dataSource: localData,
      fields: {
        id: "id",
        text: "text",
        parentId: "parent"
      }
    });
  });
}
```

### Selection

You can select a specific node by setting index value in [selectedNode](#) property or passing node's id/element to [selectNode](#) method.

To get the selected status of a given TreeView node you have to use [isSelected](#) method.

The [nodeClick](#) event will be triggered whenever TreeView node is clicked. The [beforeSelect](#) event will be triggered before the TreeView node is selected.

The [nodeSelect](#)/[nodeUnselect](#) events will be triggered based on the TreeView node click operations.

### JS

```
// Initialize and bind the TreeView with selectedNode property
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
  $(function () {
    var tree = new ej.TreeView($("#treeView"), {
      selectedNode: 2,
      dataSource: localData,
      fields: {
        id: "id",
        text: "text",
        parentId: "parent"
      }
    });
    //create an instance from an existing TreeView.
    //to select node
    tree.selectNode("4");
  });
}
```

### Ensure Visibility

To get the visibility status of the given TreeView node, you can use [isVisible](#) method.

You can ensure the specified tree node is visible by using [ensureVisible](#) method, which expands tree nodes and scrolls the TreeView control as necessary.

Also you can use [getVisibleNodes](#) method to get currently visible nodes in TreeView.

**JS**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
    var localData = [
        { id: 1, text: "Item 1" },
        { id: 2, text: "Item 2" },
        { id: 3, text: "Item 3" },
        { id: 4, text: "Item 4" },
        { id: 5, parent: 1, text: "Item 1.1" },
        { id: 6, parent: 1, text: "Item 1.2" },
        { id: 7, parent: 1, text: "Item 1.3" },
        { id: 8, parent: 3, text: "Item 3.1" },
        { id: 9, parent: 3, text: "Item 3.2" },
        { id: 10, parent: 5, text: "Item 1.1.1" }
    ];
    $(function () {
        var tree = new ej.TreeView($("#treeView"), {
            fields: { dataSource: localData, id: "id", parentId: "parent", text:
                "text" }
        });
    });
    function onClick() {
        //create an instance from an existing TreeView.
        tree.ensureVisible("10");
    }
}

```

**Checkboxes**

TreeView consists of in-built checkbox option and it can be displayed to the left of the tree node by setting the [showCheckbox](#) property as true. It allows you to select more than one node at a time.

**Indeterminate Checkboxes**

TreeView supports tristate checkboxes in addition with standard two state checkboxes. By default TreeView has been enabled with tristate in checkboxes. You can enable 2-state checkboxes by using [autoCheckParentNode](#) as false.

**HTML**

```

<!--create the TreeView wrapper-->
<div id="treeView"> </div>

```

**JS**

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
    $(function () {
        var tree = new ej.TreeView($("#treeView"), {
            showCheckbox: true, // shows each node with checkboxes
            autoCheckParentNode: false, // enable 2-state checkboxes
            fields: {
                dataSource: [

```

```

text: "Item 1",
child: [
  { text: "Item 1.1" },
  { text: "Item 1.2" }
],
{ text: "Item 2" },
{ text: "Item 3" }
]
}
});
});
}

```

### Auto Checkable

By default checkbox state of child nodes depends up on parent node checkbox state and also parent node state gets updated based on child nodes state. You can turn off this option by setting [autoCheck](#) as false to make independent parent and child nodes checkboxes.

### HTML

```

<!--create the TreeView wrapper-->
<div id="treeView"> </div>

```

### JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
  $(function () {
    var tree = new ej.TreeView($("#treeView"), {
      showCheckbox: true, // shows each node with checkboxes
      autoCheck: false, // turns off auto check of parent and child nodes
      fields: {
        dataSource: [
          {
            text: "Item 1",
            child: [
              { text: "Item 1.1" },
              { text: "Item 1.2" }
            ]
          },
          { text: "Item 2" },
          { text: "Item 3" }
        ]
      }
    });
  });
}

```

### Check or Uncheck Node

Tree node can be checked or unchecked using [checkNode](#) and [uncheckedNode](#) methods while [showCheckbox](#) property is enabled in TreeView. Also you can get or set the checked nodes of TreeView

using [checkedNodes](#) property, which indicates the checked nodes index collection as array. The [nodeCheck](#) and [nodeUncheck](#) event occurs based on checkbox state.

You can use [isNodeChecked](#) method to check the particular TreeView node is checked or unchecked. Also you can use [checkAll](#) method to check all the nodes in TreeView.

### JS

```
//to check node
tree.checkNode("2");
//to uncheck node
tree.uncheckNode("2");
```

### Get Checked Nodes

To get checked nodes of TreeView, you can use [getCheckedNodes](#) method. It returns the collection of tree node.

Also you can get currently checked nodes indexes in TreeView by using [getCheckedNodesIndex](#) method.

### HTML

```
<!--create the TreeView wrapper-->
<div id="treeView"> </div>
```

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
    $(function () {
        var tree = new ej.TreeView($("#treeView"), {
            showCheckbox: true, // shows each node with checkboxes
            autoCheck: false, // turns off auto check of parent and child nodes
            fields: {
                dataSource: [
                    {
                        text: "Item 1",
                        child: [
                            { text: "Item 1.1" },
                            { text: "Item 1.2" }
                        ]
                    },
                    { text: "Item 2" },
                    { text: "Item 3" }
                ]
            }
        });
        function onClick() {
            //to get checked nodes
            tree.getCheckedNodes();
        }
    });
}
```

## Drag and drop

To perform drag and drop operation in TreeView, specify [allowDragAndDrop](#) as true. It allows you to drag and drop node in all level of same TreeView.

### JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
    var localData = [
        { id: 1, text: "Item 1" },
        { id: 2, text: "Item 2" },
        { id: 3, text: "Item 3" },
        { id: 4, text: "Item 4" },
        { id: 5, parent: 1, text: "Item 1.1" },
        { id: 6, parent: 1, text: "Item 1.2" },
        { id: 7, parent: 1, text: "Item 1.3" },
        { id: 8, parent: 3, text: "Item 3.1" },
        { id: 9, parent: 3, text: "Item 3.2" },
        { id: 10, parent: 5, text: "Item 1.1.1" }
    ];
    $(function () {
        var tree = new ej.TreeView($("#treeView"), {
            allowDragAndDrop: true,
            fields: { dataSource: localData, id: "id", parentId: "parent", text:
                "text" }
        });
    });
}

```

**Note:** TreeView provides much easier option to drop the dragged nodes at any levels by indicator lines with icons.

### Position Indicators

TreeView provides much easier option to drop the dragged nodes at any levels by indicator lines with icons such as line, plus/minus and restrict icons while dragging and dropping the nodes. It represents exact position where the node to be dropped as sibling or child. Refer below screen shot of position indicator.

| Indicators               | description  |
|--------------------------|--|
| Plus icon                | represents the dragged node to be added as child of targeted node          |
| Minus with restrict icon | represents the dragged node not to be dropped at the hovered region        |
| In between icon          | represents the dragged node to be dropped in between the nodes as siblings |

### Restriction

You can restrict the dragged nodes to be dropped at siblings or children's level by using [allowDropSibling](#) and [allowDropChild](#) property.

### JS



```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
  var localData = [
    { id: 1, text: "Item 1" },
    { id: 2, text: "Item 2" },
    { id: 3, text: "Item 3" },
    { id: 4, text: "Item 4" },
    { id: 5, parent: 1, text: "Item 1.1" },
    { id: 6, parent: 1, text: "Item 1.2" },
    { id: 7, parent: 1, text: "Item 1.3" },
    { id: 8, parent: 3, text: "Item 3.1" },
    { id: 9, parent: 3, text: "Item 3.2" },
    { id: 10, parent: 5, text: "Item 1.1.1" }
  ];
  $(function () {
    var tree = new ej.TreeView($("#treeView"), {
      allowDragAndDrop: true,
      allowDropSibling: true, // allows to drop sibling
      allowDropChild: false, // restricts to drop as child
      fields: { dataSource: localData, id: "id", parentId: "parent", text:
        "text" }
    });
  });
}

```

### Drag and Drop between Trees

You can drag and drop tree nodes between two TreeView by setting [allowDragAndDrop] You can drag and drop tree nodes between two TreeView by setting [allowDragAndDrop](#) as true along with [allowDragAndDropAcrossControl](#) as true.

The [nodeDrag](#), [nodeDragStart](#), [nodeDragStop](#) and

[nodeDropped](#) event occurs based on Treeview node drag and drop state.

### JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
  var tree1 = [
    { id: 1, text: "Item 1" },
    { id: 2, text: "Item 2" },
    { id: 3, text: "Item 3" },
    { id: 4, text: "Item 4" },
    { id: 5, parent: 1, text: "Item 1.1" },
    { id: 6, parent: 1, text: "Item 1.2" },
    { id: 7, parent: 1, text: "Item 1.3" },
    { id: 8, parent: 3, text: "Item 3.1" },
    { id: 9, parent: 3, text: "Item 3.2" },
    { id: 10, parent: 5, text: "Item 1.1.1" }
  ];
  var tree2 = [
    { id: 11, text: "Item 5" },
    { id: 12, text: "Item 6" },
    { id: 13, text: "Item 7" },
  ];
}

```

```

{ id: 14, text: "Item 4" },
{ id: 15, parent: 11, text: "Item 5.1" },
{ id: 16, parent: 11, text: "Item 5.2" },
{ id: 17, parent: 11, text: "Item 5.3" },
{ id: 18, parent: 13, text: "Item 7.1" },
{ id: 19, parent: 13, text: "Item 7.2" },
{ id: 10, parent: 15, text: "Item 5.1.1" }
];
$(function () {
var tree = new ej.TreeView($("#treeViewDrag"), {
allowDragAndDrop: true,
allowDragAndDropAccessControl: true,
allowDropSibling: true, // allows to drop sibling
allowDropChild: true, // allows to drop as child
fields: { dataSource: tree1, id: "id", parentId: "parent", text: "text" }
});
var tree = new ej.TreeView($("#treeViewDrop"), {
allowDragAndDrop: true,
allowDragAndDropAccessControl: true,
allowDropSibling: true, // allows to drop sibling
allowDropChild: true, // allows to drop as child
fields: { dataSource: tree2, id: "id", parentId: "parent", text: "text" }
});
});
}

```

For more details about drag and drop between TreeView, refer the sample [here](#).

### Auto Node Structuring

You may not need to have two TreeView to be in same structured node while drag and drop the nodes between them. But after the node has been dropped, it should get structure of the TreeView node in which dropped. By default TreeView auto structure the node whenever you drop a node from different tree.

### JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
var tree1 = [
{ id: 1, text: "Item 1" },
{ id: 2, text: "Item 2" },
{ id: 3, text: "Item 3" },
{ id: 4, text: "Item 4" },
{ id: 5, parent: 1, text: "Item 1.1" },
{ id: 6, parent: 1, text: "Item 1.2" },
{ id: 7, parent: 1, text: "Item 1.3" },
{ id: 8, parent: 3, text: "Item 3.1" },
{ id: 9, parent: 3, text: "Item 3.2" },
{ id: 10, parent: 5, text: "Item 1.1.1" }
];
var tree2 = [
{ id: 11, text: "Item 5" },
{ id: 12, text: "Item 6" },
{ id: 13, text: "Item 7" },
{ id: 14, text: "Item 4" },

```

```

{ id: 15, parent: 11, text: "Item 5.1" },
{ id: 16, parent: 11, text: "Item 5.2" },
{ id: 17, parent: 11, text: "Item 5.3" },
{ id: 18, parent: 13, text: "Item 7.1" },
{ id: 19, parent: 13, text: "Item 7.2" },
{ id: 10, parent: 15, text: "Item 5.1.1" }
];
$(function () {
    // initialize and bind the TreeView with local data
    $("#treeViewDrag").ejTreeView({
        allowDragAndDrop: true,
        allowDragAndDropAccessControl: true,
        allowDropSibling: true, // allows to drop sibling
        allowDropChild: true, // allows to drop as child
        fields: { dataSource: tree1, id: "id", parentId: "parent", text: "text"
        },
        showCheckbox: true
    });
    $("#treeViewDrop").ejTreeView({
        allowDragAndDrop: true,
        allowDragAndDropAccessControl: true,
        allowDropSibling: true, // allows to drop sibling
        allowDropChild: true, // allows to drop as child
        fields: { dataSource: tree2, id: "id", parentId: "parent", text: "text" }
    });
});
}

```

For more details about drag and drop between TreeView, refer the sample [here](#).

**Note:** Auto node structure applicable for well-structured node object.

## Multiple Selection

TreeView supports to select the multiple nodes by specifying [allowMultiSelection](#) as true. It allows you to select more than one nodes in TreeView.

### HTML

```

<!--create the TreeView wrapper-->
<div id="treeMultiSelect"></div>

```

### JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
    var localData = [
        { id: 1, text: "Item 1" },
        { id: 2, text: "Item 2" },
        { id: 3, text: "Item 3" },
        { id: 4, text: "Item 4" },
        { id: 5, parent: 1, text: "Item 1.1" },
        { id: 6, parent: 1, text: "Item 1.2" },
        { id: 7, parent: 1, text: "Item 1.3" },
        { id: 8, parent: 3, text: "Item 3.1" },
        { id: 9, parent: 3, text: "Item 3.2" },
    ];
}

```

```
{ id: 10, parent: 5, text: "Item 1.1.1" }
];
$(function () {
var tree = new ej.TreeView($("#treeMultiSelect"), {
allowMultiSelection: true, //enable multiple selection in TreeView
fields: { dataSource: localData, id: "id", parentId: "parent", text:
"text" }
});
});
}
```

For more details about multiple selection in TreeView, refer the sample [here](#).

### Select Nodes

To select more than one nodes of TreeView, you can use [selectedNodes](#) property. It will select the TreeView nodes from the given indexes.

### HTML

```
<!--create the TreeView wrapper-->
<div id="treeMultiSelect"></div>
```

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
var localData = [
{ id: 1, text: "Item 1" },
{ id: 2, text: "Item 2" },
{ id: 3, text: "Item 3" },
{ id: 4, text: "Item 4" },
{ id: 5, parent: 1, text: "Item 1.1" },
{ id: 6, parent: 1, text: "Item 1.2" },
{ id: 7, parent: 1, text: "Item 1.3" },
{ id: 8, parent: 3, text: "Item 3.1" },
{ id: 9, parent: 3, text: "Item 3.2" },
{ id: 10, parent: 5, text: "Item 1.1.1" }
];
$(function () {
var tree = new ej.TreeView($("#treeMultiSelect"), {
allowMultiSelection: true, //enable multiple selection in TreeView
fields: { dataSource: localData, id: "id", parentId: "parent", text:
"text" }
});
tree.option("selectedNodes", [0, 5, 6]); //select the TreeView nodes from
the given indexes
});
}
```

### Get Selected Nodes

To get the selected Nodes of TreeView, you can use [getSelectedNodes](#) method. It returns the collections of TreeView selected nodes.

You can use [getSelectedNodesIndex](#) method to get the index positions of currently selected nodes.

**HTML**

```
<!--create the TreeView wrapper-->
<div id="treeMultiSelect"></div>
```

**JS**

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
  var localData = [
    { id: 1, text: "Item 1" },
    { id: 2, text: "Item 2" },
    { id: 3, text: "Item 3" },
    { id: 4, text: "Item 4" },
    { id: 5, parent: 1, text: "Item 1.1" },
    { id: 6, parent: 1, text: "Item 1.2" },
    { id: 7, parent: 1, text: "Item 1.3" },
    { id: 8, parent: 3, text: "Item 3.1" },
    { id: 9, parent: 3, text: "Item 3.2" },
    { id: 10, parent: 5, text: "Item 1.1.1" }
  ];
  $(function () {
    // initialize and bind the TreeView with local data
    $("#treeMultiSelect").ejTreeView({
      allowMultiSelection: true, //enable multiple selection in TreeView
      fields: { dataSource: localData, id: "id", parentId: "parent", text:
        "text" }
    });
    tree.getSelectedNodes(); //returns the collections of TreeView selected
    nodes
  });
}
```

**Drag and Drop Multiple Nodes**

TreeView supports to drag and drop multiple nodes by specifying [allowMultiSelection](#) as true along with [allowDragAndDrop](#) as true. It allows you to drag and drop multiple nodes in TreeView.

**HTML**

```
<!--create the TreeView wrapper-->
<table>
<tr>
<td>
<div id="treeViewDrag"></div>
</td>
<td>
<div id="treeViewDrop"></div>
</td>
</tr>
</table>
```

**JS**

```
/// <reference path="tsfiles/jquery.d.ts" />
```

```

/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
  var tree1 = [
    { id: 1, text: "Item 1" },
    { id: 2, text: "Item 2" },
    { id: 3, text: "Item 3" },
    { id: 4, text: "Item 4" },
    { id: 5, parent: 1, text: "Item 1.1" },
    { id: 6, parent: 1, text: "Item 1.2" },
    { id: 7, parent: 1, text: "Item 1.3" },
    { id: 8, parent: 3, text: "Item 3.1" },
    { id: 9, parent: 3, text: "Item 3.2" },
    { id: 10, parent: 5, text: "Item 1.1.1" }
  ];
  var tree2 = [
    { id: 11, text: "Item 5" },
    { id: 12, text: "Item 6" },
    { id: 13, text: "Item 7" },
    { id: 14, text: "Item 4" },
    { id: 15, parent: 11, text: "Item 5.1" },
    { id: 16, parent: 11, text: "Item 5.2" },
    { id: 17, parent: 11, text: "Item 5.3" },
    { id: 18, parent: 13, text: "Item 7.1" },
    { id: 19, parent: 13, text: "Item 7.2" },
    { id: 10, parent: 15, text: "Item 5.1.1" }
  ];
  $(function () {
    var tree = new ej.TreeView($("#treeViewDrag"), {
      allowMultiSelection: true,
      allowDragAndDrop: true,
      fields: { dataSource: tree1, id: "id", parentId: "parent", text: "text" },
    });
    var tree1 = new ej.TreeView($("#treeViewDrop"), {
      allowMultiSelection: true,
      allowDragAndDrop: true,
      fields: { dataSource: tree2, id: "id", parentId: "parent", text: "text" }
    });
  });
}

```

For more details about multiple drag and drop in TreeView, refer the sample [here](#).

### Full Row Selection

The TreeView control provides the option to highlight a full row of tree view nodes. When [fullRowSelect](#) is **true**, a selection highlights the entire width of the tree view, display instead of the width of just the tree node text. It makes selection easier.

### HTML

```

<!--create the TreeView wrapper-->
<div id="fullRowTree"></div>

```

### JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
  var localData = [
    { id: 1, name: "Discover Music", hasChild: true, expanded: true },
    { id: 2, pid: 1, name: "Hot Singles", selected: true },
    { id: 3, pid: 1, name: "Rising Artists" },
    { id: 4, pid: 1, name: "Live Music" },
    { id: 6, pid: 1, name: "Best of 2013 So Far" },
    { id: 7, name: "Sales and Events", hasChild: true, expanded: true },
    { id: 8, pid: 7, name: "100 Albums - $5 Each" },
    { id: 9, pid: 7, name: "Hip-Hop and R&B Sale" },
    { id: 10, pid: 7, name: "CD Deals" },
    { id: 11, name: "Categories", hasChild: true },
    { id: 12, pid: 11, name: "Songs" },
    { id: 13, pid: 11, name: "Bestselling Albums" },
    { id: 14, pid: 11, name: "New Releases" },
    { id: 15, pid: 11, name: "Bestselling Songs" },
    { id: 16, name: "MP3 Albums", hasChild: true },
    { id: 17, pid: 16, name: "Rock" },
    { id: 18, pid: 16, name: "Gospel" },
    { id: 19, pid: 16, name: "Latin Music" },
    { id: 20, pid: 16, name: "Jazz" },
    { id: 21, name: "More in Music", hasChild: true },
    { id: 22, pid: 21, name: "Music Trade-In" },
    { id: 23, pid: 21, name: "Redeem a Gift Card" },
    { id: 24, pid: 21, name: "Band T-Shirts" },
    { id: 25, pid: 21, name: "Mobile MVC" }
  ];
  $(function () {
    var tree = new ej.TreeView($("#fullRowTree"), {
      fullRowSelect: true, //enable full row selection in TreeView
      fields: {
        dataSource: localData, id: "id", parentId: "pid", text: "name",
        hasChild: "hasChild", expanded: "expanded", selected: "selected"
      }
    });
  });
}

```

For more details about full row selection in TreeView, refer the sample [here](#).

## Template

Tree nodes can be customized by using [template](#) property. TreeView template option requires addition JS library called 'JsRender', which helps to create the structured way of tree nodes with simple codes and increased performance. To know more about JsRender - <http://www.jsviews.com/>.

## JS

```

<script id="treeTemplate" type="text/x-jsrender">
  {{"{{"}}"}}if hasChild{{{}}}
  <div class={{"{{"}}"}}>className{{{}}}{{"{{"}}"}}>name{{{}}}</div>
  {{"{{"}}"}}else{{{}}}
  <div class="cont-list">
    imgId{{{}}}.png" />
  </div>
</script>

```

```

<div class="cont-del"></div>
<b>{{"{{"}}>name{{}}}}</b><br />
<span>{{"{{"}}>city{{}}}}</span>
<br />
<span>{{"{{"}}>phone{{}}}}</span>
</div>
<div class="treeFooter"></div>
</div>
{{"{{"}}/if{{}}}}
</script>

```

## JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
$(function () {
var treeData = [
{ id: 1, name: "UK", className: "uk-style", hasChild: true, expanded:
true },
{ id: 2, pid: 1, imgId: "1", name: "Steven John", city: "London", phone:
"555-5665-2323" },
{ id: 3, name: "USA", className: "usa-style", hasChild: true, expanded:
true },
{ id: 5, pid: 3, imgId: "3", name: "Andrew", city: "Capital way", phone:
"934-8374-2455" },
{ id: 4, pid: 3, imgId: "2", name: "Angelica", city: "Dayton", phone:
"988-4243-0806" }
];
var tree = new ej.TreeView($("#treeView"), {
fields: {
dataSource: treeData,
id: "id", parentId: "pid", text: "name", hasChild: "hasChild"
},
template: "#treeTemplate"
});
});
}

```

For more details about node template, refer the sample [here](#).

## State Persistence

TreeView state can be persisted by using [enablePersistence](#). In which entire modal has been persisted excluding data source in order to maintain performance.

The model values of below are maintained through Id basis of tree node.

- **selected**
- **checked**
- **expanded/collapsed state**

---

**Note:** "UI-li" template option, state has been persisted by index.

---



TreeView stores its model in local storage / cookies of browser before page refreshes and reinitialized with their stored model after refresh.

### JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module TreeViewComponent {
  var localData = [
    { id: 1, text: "Item 1" },
    { id: 2, text: "Item 2" },
    { id: 3, text: "Item 3" },
    { id: 4, text: "Item 4" },
    { id: 5, parent: 1, text: "Item 1.1" },
    { id: 6, parent: 1, text: "Item 1.2" },
    { id: 7, parent: 1, text: "Item 1.3" },
    { id: 8, parent: 3, text: "Item 3.1" },
    { id: 9, parent: 3, text: "Item 3.2" },
    { id: 10, parent: 5, text: "Item 1.1.1" }
  ];
  $(function () {
    // initialize and bind the TreeView with local data
    $("#treeView").ejTreeView({
      enablePersistence: true,
      fields: { dataSource: localData, id: "id", parentId: "parent", text:
        "text" }
    });
  });
}

```

### Keyboard Interaction

TreeView support all possible keyboard interaction and it can be allowed by specifying [allowKeyboardNavigation](#) as true.

You can use the keyboard short cut keys as an alternative to the mouse and interact with TreeView.

Keyboard shortcut keys are,

| Key    | Description   |
|--------|---|
| F2     | To edit the selected node                               |
| Delete | To delete the selected node                             |
| Down   | To hover the next index of selected node in downwards   |
| Up     | To hover the previous index of selected node in upwards |
| Right  | To expands the node                                     |
| Left   | To collapse the node                                    |

|       |  |
|-------|--|
| Enter | To select the node                     |
| Space | To check or uncheck node               |
| Home  | Hovers the first root node of TreeView |
| End   | Hovers the last node of the TreeView   |
| Esc   | Focus out the edit state               |

For more details about keyboard support, refer the sample [here](#).

### How To

#### Update the modified data from tree to database

TreeView allow you to get the updated tree data after performing such operation like node editing, drag and drop, add and remove node. Using [getTreeData](#) method you can get the updated tree data.

Refer the sample from the link [Updated tree data](#) to know how to get updated data from TreeView.

You can also get the updated data source for remote data binding after performing the operation like editing, selecting/unselecting, expanding/collapsing, checking/unchecking and removing node. You cannot get the updated data source when you perform operation like drag and drop, adding node for remote data binding.

The updated data source also contains custom attributes ("ContactTitle", "OrderID", "EmployeeID", "Freight") if you return these from server.

Refer the sample from the [link](#) to know more about how to get updated data with custom attributes from TreeView for remote data binding.

#### TreeView context menu to process node operations

TreeView allow you to perform some tree operation like add, remove, update, move, check, uncheck, select node dynamically by using built-in methods and properties. Using those methods and properties, you can trigger it through context menu item click action to manipulate node.

Refer the sample from the link [TreeContextMenu](#) to know how to use context menu to perform tree operations.

#### Sorted data using refresh method

TreeView allow you to refresh the entire tree data by using [refresh](#) method. Refer the sample from the link [Sorted tree data](#) to know how to update entire tree data.

#### Persist updated data after edit, add and remove node

TreeView allow you to persist the updated data after performing node manipulation. By using persistence option, you can sustain on page refresh.

The [nodeAdd](#), [nodeCut](#), [nodeDelete](#) and [nodePaste](#) events occurs based on Treeview node manipulation. The [beforeAdd](#),

[beforeCut](#), [beforeDelete](#) and [beforePaste](#) events are triggered before the TreeView component node manipulation.

Refer the sample from the link [PersistData](#) to know how to persist updated tree data after refresh.

#### Filtering nodes in TreeView

You can able to filter TreeView nodes based on node text. Refer the sample from the link [FilterTreeNodes](#) to know how to filter tree nodes based on the node text.

#### AngularJS data binding to update data while add and remove node

TreeView allow you to bind and update tree data in mapped data component while add and removing node using AngularJS binding. Refer the sample from the link [AngularDataBinding](#) to know how to update data using AngularJS binding.

#### Set Tooltip for TreeView nodes

TreeView allow you to set tooltip option to TreeView nodes by using [fields.linkAttribute](#) property of TreeView. Refer the sample from the link [Tooltip](#) to know how to set tooltip for TreeView nodes.

#### Auto hide/show the expand/collapse icon of TreeView

You can able to display expand icon on mouse entering into TreeView and hide while leaving from the TreeView. Refer the sample [AutoHide](#) to know how to hide/show the expand/collapse icons automatically based on mouse position.

#### Customize the expand/collapse icons of TreeView

You can able to customize the TreeView expand and collapse icon by using “cssClass” property of TreeView. Refer the sample [CustomizeIcons](#) to know how to customize the expand/collapse icons.

## Uploadbox

### Overview

The **Essential JavaScript Uploadbox** control supports uploading files into the designated server, regardless of the file format and size. The **Uploadbox** control helps you with the selection of files to upload to the server. The **Uploadbox** control has theme support.

### Key Features

- **Modern look:** Rich appearance with theme support.
- **RTL:** Supports right-to-left alignment.
- **Auto Upload:** Support for manual and automatic uploading of files is included.
- **Sync/Async Upload:** Supports synchronous and asynchronous uploading of files.
- **Multiple files:** Supports uploading multiple files.

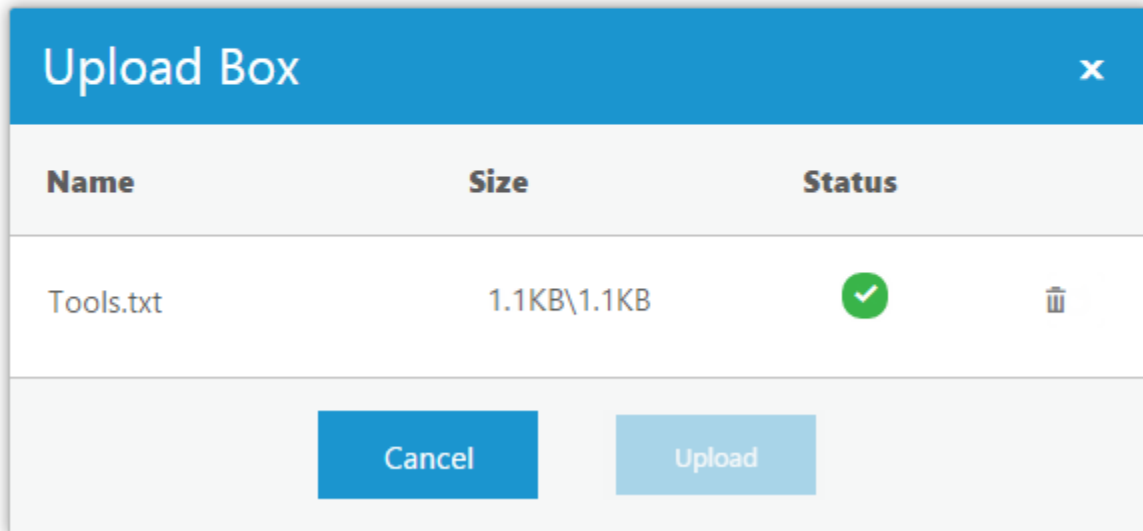
### Getting Started

This section explains briefly about how to create an **Uploadbox** in your application with **Typescript**. **Essential JavaScript Uploadbox** widget provides support to upload files or photos within your web page. From the following guidelines, you can learn how to upload the files that are used in a Resume Upload scenario. This helps you to restrict some file extensions when you upload the resume in server by using **Uploadbox** control.

The following screenshot demonstrates the functionality of **Uploadbox** with the file extension.

Extensions:

|                                     |                                      |                                       |
|-------------------------------------|--------------------------------------|---------------------------------------|
| <input type="text" value=".txt"/>   | <input type="button" value="Allow"/> | <input type="button" value="Browse"/> |
| <input type="text" value="Format"/> | <input type="button" value="Deny"/>  |                                       |



In the above screenshot, you can upload a resume that allows **.txt** files. This helps you to avoid unsupported resume formats getting uploaded in a server.

**Note:** To get upload the file, you should either run this sample in Visual Studio IDE or host in local IIS.

### Create Uploadbox in Typescript

Create an **HTML** page and add the scripts references in the order mentioned in the following code example.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/bootstrap-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.0.0.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="app.js"></script>
</head>
<body>
</body>
</html>
```

#### HTML

```
<div id="targetElement">
<div id="UploadDefault"></div>
</div>
```

Add the given styles to display the **Uploadbox** with margin alignments.

### CSS

```
<style>
#targetElement {
width: 500px;
height: 500px;
margin: 0 auto;
}
#UploadDefault {
margin: 0 auto;
}
</style>
```

Create a new handler file (.ashx) and save it as **saveFiles.ashx** and then copy the following code into it.

### C#

```
SaveFiles.ashx
public void ProcessRequest(HttpContext context)
{
    string targetFolder = HttpContext.Current.Server.MapPath("uploadfiles");
    if (!Directory.Exists(targetFolder))
    {
        Directory.CreateDirectory(targetFolder);
    }
    HttpRequest request = context.Request;
    HttpFileCollection uploadedFiles = context.Request.Files;
    if (uploadedFiles != null && uploadedFiles.Count > 0)
    {
        for (int i = 0; i < uploadedFiles.Count; i++)
        {
            if (uploadedFiles[i].FileName != null && uploadedFiles[i].FileName != "")
            {
                string fileName = uploadedFiles[i].FileName;
                int indx = fileName.LastIndexOf("\\");
                if (indx > -1)
                {
                    fileName = fileName.Substring(indx + 1);
                }
                uploadedFiles[i].SaveAs(targetFolder + "\\\" + fileName);
            }
        }
    }
}
```

Create a new handler file (.ashx) and save it as **removeFiles.ashx** and then copy the following code into it.

### C#

```
removeFiles.ashx
public void ProcessRequest(HttpContext context)
{
```

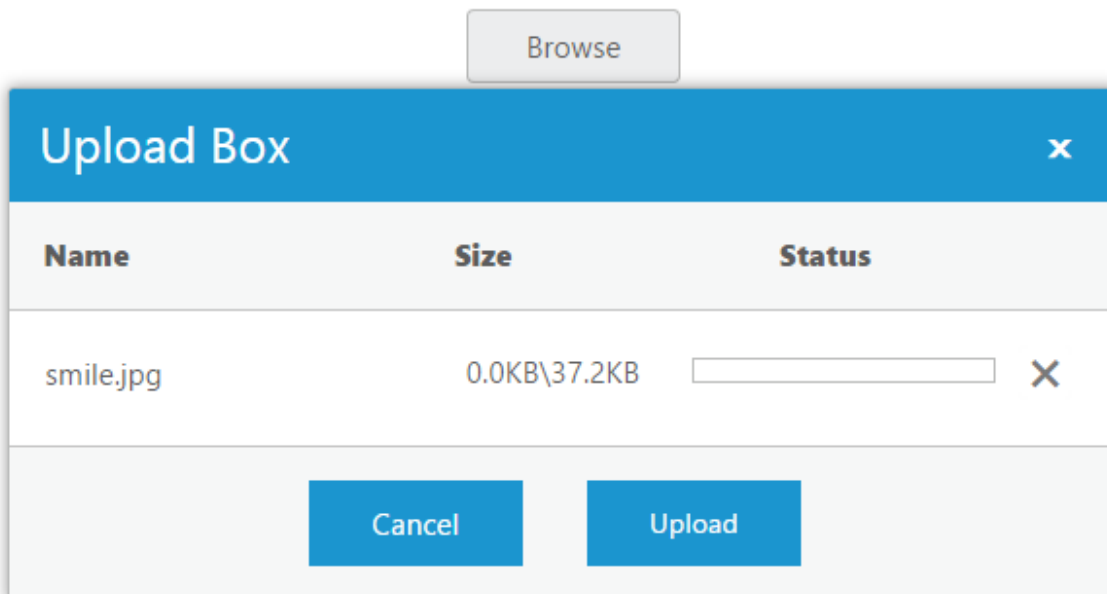
```
System.Collections.Specialized.NameValueCollection s =
context.Request.Params;
string fileName = s["fileNames"];
string targetFolder = HttpContext.Current.Server.MapPath("uploadfiles");
if (Directory.Exists(targetFolder))
{
string physicalPath = targetFolder + "\\\" + fileName;
if (System.IO.File.Exists(physicalPath))
{
System.IO.File.Delete(physicalPath);
}
}
}
```

Add the following code example to assign saveUrl and removeUrl for Uploadbox

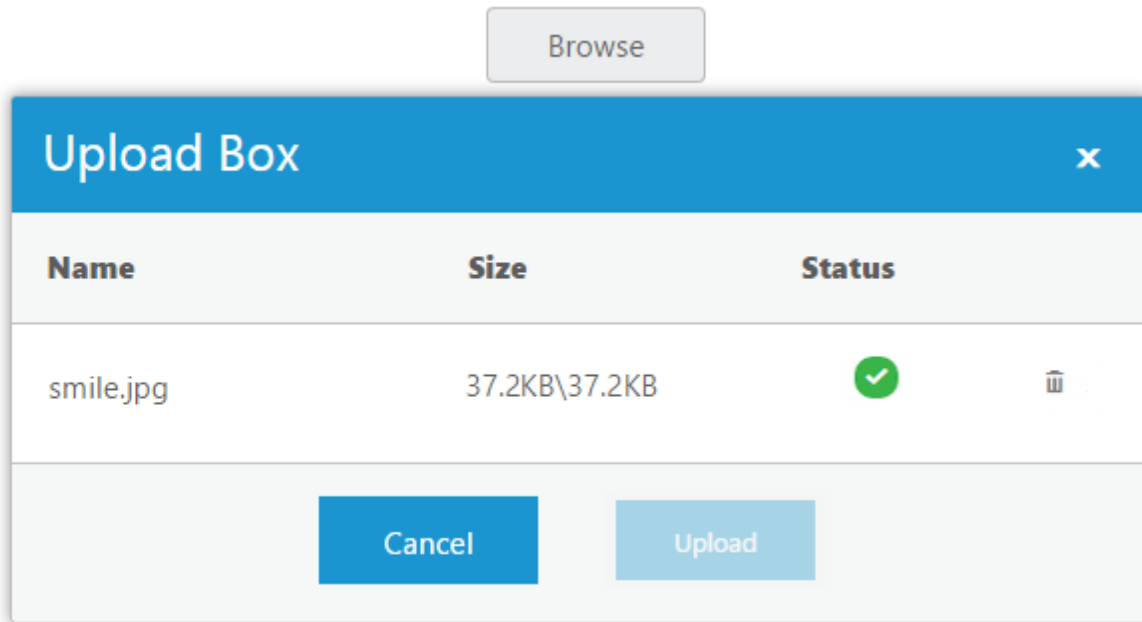
#### HTML

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module UploadComponent {
$(function () {
var sample = new ej.Uploadbox($("#UploadDefault"), {
saveUrl: "saveFiles.ashx",
removeUrl: "removeFiles.ashx"});
});
}
```

The following screenshot displays an **Uploadbox** control.



After you upload the files, the following screen shot is displayed.



**Note:** The above screenshot displays the Uploadbox control that shows the files are uploaded successfully.

#### Set Restriction for File Extension

In a real-time scenario, some file extensions are restricted. You can allow files and restrict files by using the following two properties **extensionsAllow** and **extensionsDeny** enabled in **Uploadbox**.

**Note:** The SaveUrl and RemoveUrl are the same as above (see step 4)

Add input elements to create elements for file extension.

**Note:** Add the following input elements and two button elements to give file extensions that should support uploading.

#### HTML

```
<div id="targetElement">
  <table id="uploadTable">
    <tr>
      <td>
        Extensions:
      </td>
    </tr>
    <tr>
      <td>
        <input type="text" id="fileallow" class="ejinputtext"
        placeholder="Format" />
        <input type="button" class="e-btn" id="upbutton1" value="Allow" />
      </td>
    </tr>
  </table>
</div>
```

```

<input type="text" id="filedeny" class="ejinputtext" placeholder="Format"
/>
<input type="button" class="e-btn" id="upbutton2" value="Deny" />
</td>
<td>
<div id="UploadDefault"></div>
</td>
</tr>
</table>
</div>

```

Add the following code example for click event

### HTML

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module UploadComponent {
$(function () {
var sample = new ej.Uploadbox($("#UploadDefault"), {
saveUrl: "saveFiles.ashx",
removeUrl: "removeFiles.ashx"
});
var allowButton = new ej.Button($("#upbutton1"), {
size: "normal", text: "Allow", click: function () {
var uploadobject = $("#UploadDefault").data("ejUploadbox");
uploadobject.option('extensionsAllow', $("#fileallow").val());
uploadobject.option('extensionsDeny', "");
}
});
var denyButton = new ej.Button($("#upbutton2"), {
size: "normal", text: "Deny", click: function () {
var uploadobject = $("#UploadDefault").data("ejUploadbox");
uploadobject.option('extensionsAllow', "");
uploadobject.option('extensionsDeny', $("#filedeny").val());
}
});
});
}

```

Add the given styles to display the **Uploadbox** with margin alignments.

### CSS

```

<style>
#targetElement {
width: 520px;
height: 500px;
margin: 0 auto;
}
#UploadDefault {
float: right;
}
#uploadTable {
width: 100%;
}

```



```
#fileallow, #filedeny {  
width: 150px;  
height: 20px;  
padding: 5px;  
}  
</style>
```


**Note:** You can restrict one or more files at a time by giving it as .html,.txt

The following screenshot displays an **Uploadbox** control with the file extension.

Extensions:

Upload Box ×

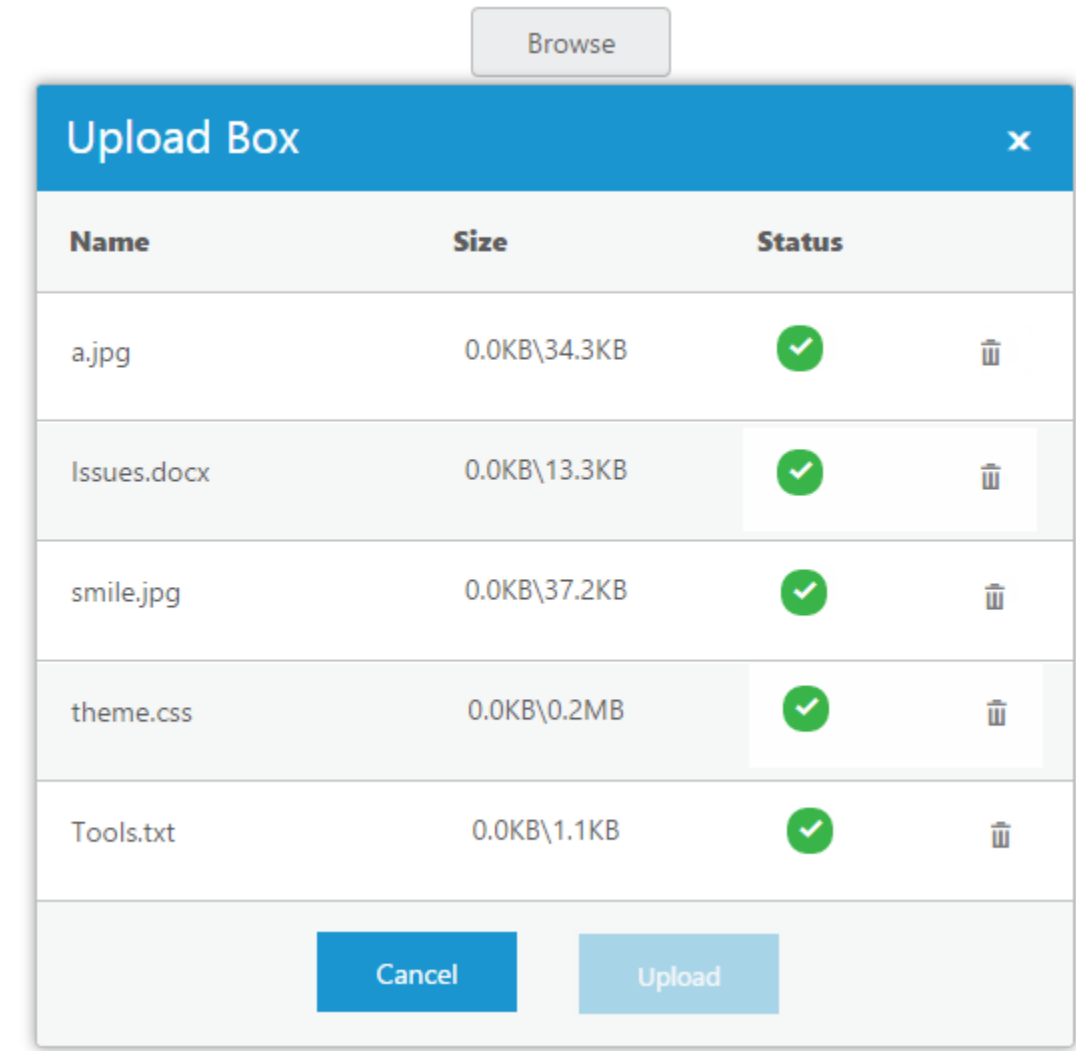
| Name      | Size        | Status  |
|-----------|-------------|---|
| Tools.txt | 1.1KB\1.1KB | <input checked="" type="checkbox"/>  |

The above screenshot shows the **Uploadbox** that allows “.txt” file formats. You can give number of file formats in both allow and deny textbox elements.

#### Upload Multiple Files

You can click the **Browse** button and select files to upload multiple files in **Uploadbox** control. You can see the selected files in **Uploadbox** control and you can upload all the files.

The following screenshot displays an **Uploadbox** control with multiple files.



## File Actions

### Save File Action

To save the uploaded file in **JS**, create the handler class and trigger the same in **saveUrl** property. In that handler, save and specify the target location for uploaded files. The data type is **string**.

The following steps explain the configuration of **saveUrl** property in the **Uploadbox**.

In the **HTML** page, add the **<div>** element to configure the **Uploadbox** element.

### HTML

```
<div id="Uploadbox"></div>
```

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
    $(function () {
        var sample = new ej.Uploadbox($("#Uploadbox"), {
```

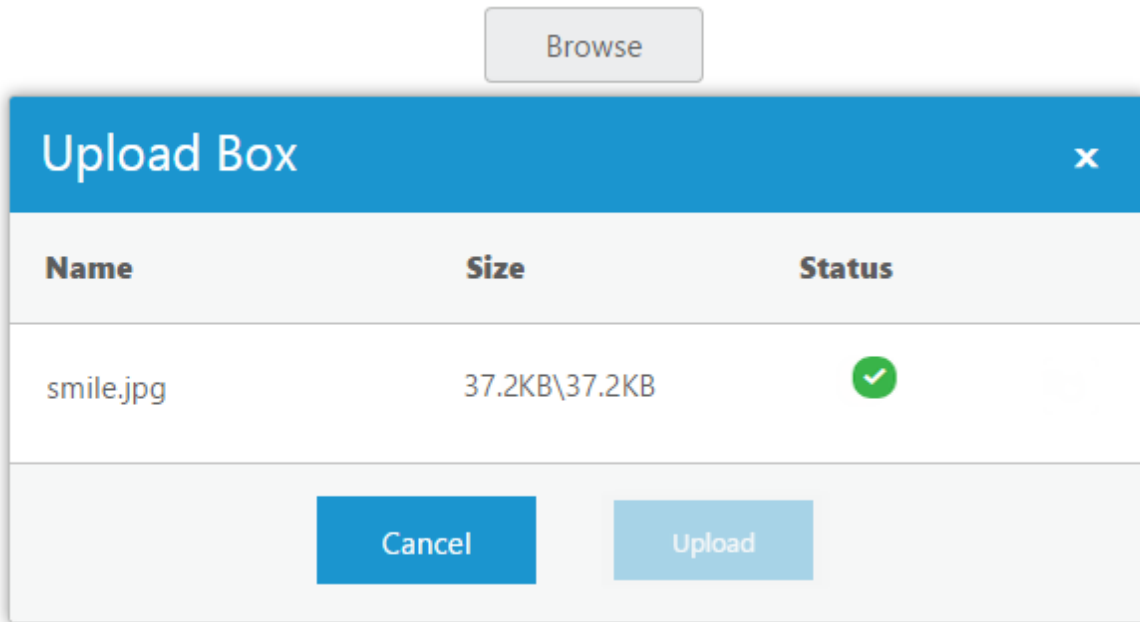
```
saveUrl: "saveFiles.ashx"
});
});
}
```

Configure the handler to save the file. Create a folder (for example, upload files) and save the uploaded files into this folder.

### **C#**

```
saveFiles.ashx
public class saveFiles : IHttpHandler {
public void ProcessRequest(HttpContext context)
{
string targetFolder = HttpContext.Current.Server.MapPath("upload-files");
if (!Directory.Exists(targetFolder))
{
Directory.CreateDirectory(targetFolder);
}
HttpRequest request = context.Request;
HttpFileCollection uploadedFiles = context.Request.Files;
if (uploadedFiles != null && uploadedFiles.Count > 0)
{
for (int i = 0; i < uploadedFiles.Count; i++)
{
string fileName = uploadedFiles[i].FileName;
int index = fileName.LastIndexOf("\\");
if (index > -1)
{
fileName = fileName.Substring(index + 1);
}
uploadedFiles[i].SaveAs(targetFolder + "\\ " + fileName);
}
}
}
public bool IsReusable
{
get
{
return false;
}
}
}
```

The following screenshot displays the output.



#### Remove File Action

To **remove** the uploaded file in **JS**, create a handler class and trigger the same in **removeUrl** property. The uploaded file has to be removed from the handler where the file is saved. This is achieved by clicking **remove** button on **upload** dialog. The data type is **string**.

The following steps explain the configuration of **removeUrl** property in **Uploadbox**.

In the **HTML** page, add the **<div>** element to configure the **Uploadbox** element.

#### HTML

```
<div id="Uploadbox"></div>
```

#### JS

```
// Initialize the control in JavaScript.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
    $(function () {
        var sample = new ej.Uploadbox($("#Uploadbox"), {
            saveUrl: "saveFiles.ashx",
            removeUrl: "removeFiles.ashx"
        });
    });
}
```

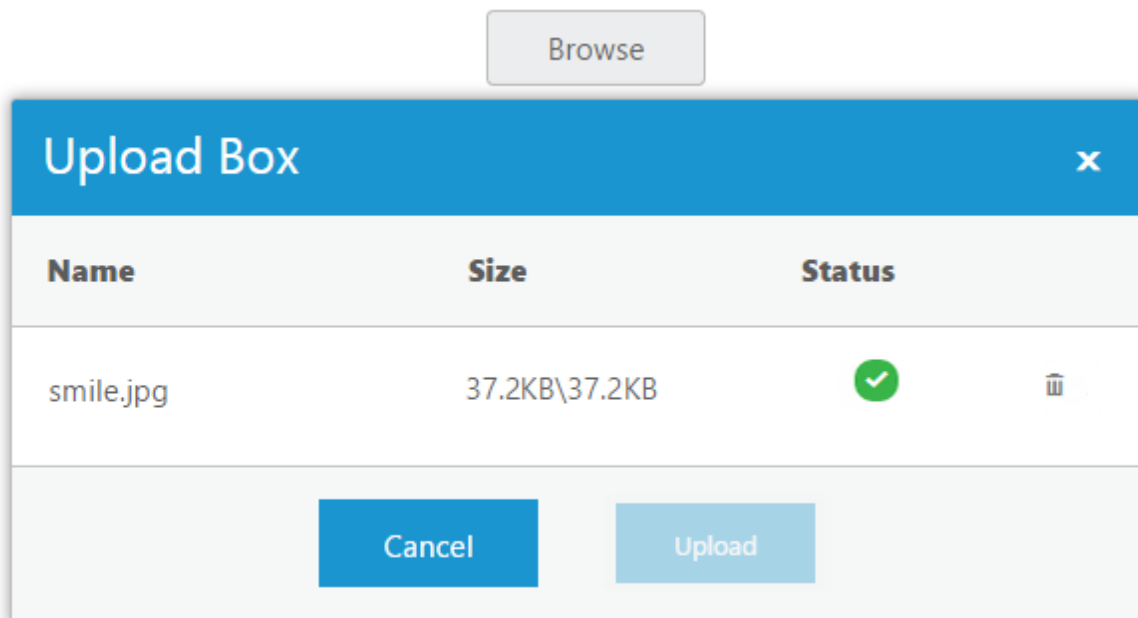
Configure the handlers to remove the file from the target location. From that location, the file is searched and removed from the '**upload-files**' folder.

#### C#

```
removeFiles.ashx
```

```
public class removeFiles : IHttpHandler
{
    public void ProcessRequest(HttpContext context)
    {
        System.Collections.Specialized.NameValueCollection s =
        context.Request.Params;
        string fileName = s["fileNames"];
        string targetFolder = HttpContext.Current.Server.MapPath("upload-files");
        if (!Directory.Exists(targetFolder))
        {
            Directory.CreateDirectory(targetFolder);
        }
        string physicalPath = targetFolder + "\\\" + fileName;
        if (System.IO.File.Exists(physicalPath))
        {
            System.IO.File.Delete(physicalPath);
        }
    }
    public bool IsReusable
    {
        get
        {
            return false;
        }
    }
}
```

The following screenshot displays the output.



#### Auto Upload

The **Uploadbox** widget provides support to upload the file automatically once file is selected by using browse button, that is, without clicking upload button. To achieve this, set the **autoUpload** property to

'true'. The data type is **Boolean**. By default, the value is set to 'false', so **upload** button is clicked to upload the files.

The following steps explain the configuration of **autoUpload** property in **Uploadbox**

In the **HTML** page, add the **<div>** element to configure the **Uploadbox** element.

#### HTML

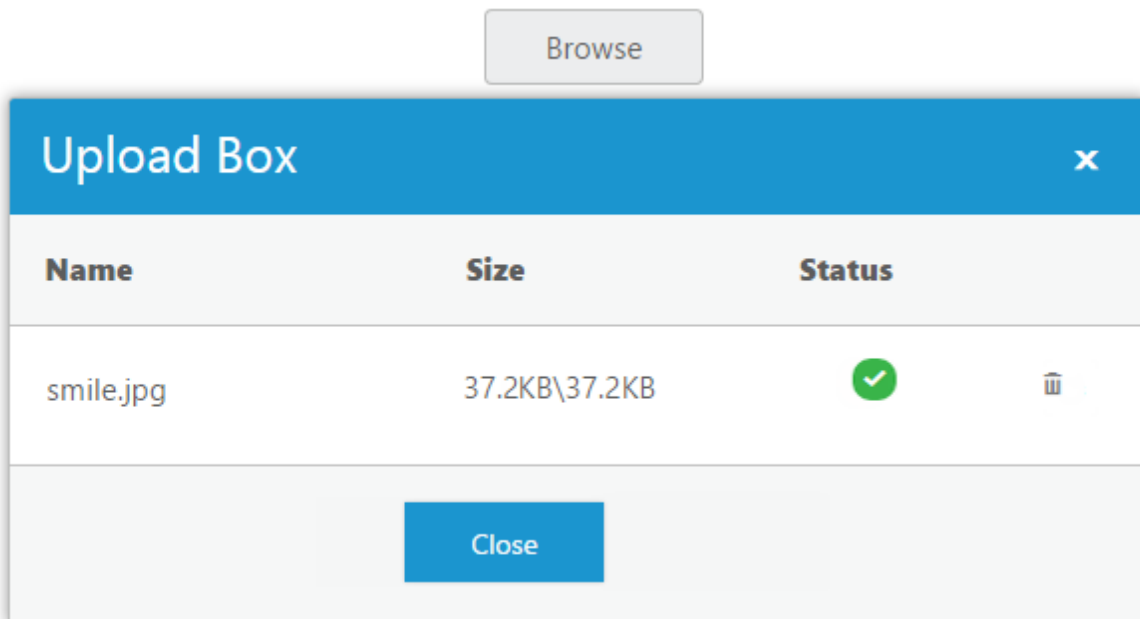
```
<div id="Uploadbox"></div>
```

#### JS

```
// Initialize the control in JavaScript.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
$(function () {
var sample = new ej.Uploadbox($("#Uploadbox"), {
saveUrl: "saveFiles.ashx",
removeUrl: "removeFiles.ashx",
autoUpload: true
});
});
}
```

For **JS**, configure **saveFiles.ashx** and **removeFiles.ashx** files as mentioned in the Save file action and Remove file action respectively.

The following screenshot displays the output.



## Restricting uploading files based on its extension

### Allow Extension

Files are filtered before they are uploaded. You can select the files to be filtered by using **browse** button. The **extensionsAllow** property allows upload of the selected extensions only. You can give multiple extensions by using comma (,). The data type is **string**.

---

**Note:** Prepend dot (.) symbol with extension like “.pdf”.

---

The following steps explain the configuration of **extensionsAllow** property in **Uploadbox**.

In the **HTML** page, add the **<div>** element to configure the **Uploadbox** element.

#### HTML

```
<div id="Uploadbox"></div>
```

#### JS

```
// Initializes the control in JavaScript.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
  $(function () {
    var sample = new ej.Uploadbox($("#Uploadbox"), {
      saveUrl: "saveFiles.ashx",
      removeUrl: "removeFiles.ashx",
      extensionsAllow: ".docx, .pdf"
    });
  });
}
```

For **JS**, configure **saveFiles.ashx** and **removeFiles.ashx** files as mentioned in the Save file action and Remove file action respectively.

### Deny Extension

Files are filtered before they are uploaded. You can select the files to be filtered by using **browse** button. The **extensionsDeny** property denies upload of the selected extensions. You can give multiple extensions by using comma (,). The data type is **string**.

---

**Note:** Prepend dot (.) symbol with extension like “.pdf”.

---

The following steps explain the configuration of **extensionsDeny** property in **Uploadbox**

In the **HTML** page, add the **<div>** element to configure the **Uploadbox** element.

#### HTML

```
<div id="Uploadbox"></div>
```

#### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
  $(function () {
    var sample = new ej.Uploadbox($("#Uploadbox"), {
```

```
saveUrl: "saveFiles.ashx",
removeUrl: "removeFiles.ashx",
extensionsDeny: ".docx, .pdf"
});
});
}
```

For **JS**, configure **saveFiles.ashx** and **removeFiles.ashx** files as mentioned in the Save file action and Remove file action respectively.

**Note:** When **extensionsDeny** and **extensionsAllow** properties have same file extension, the extension will be allowed.

## File Size

### Maximum File Size for Uploadbox

In the **Uploadbox** control, you can browse files with the size going up to gigabytes. You can restrict the files from being browsed using the **fileSize** property. When you do not use this property, it takes a default size, 31457280B, that is, 31MB. When this size exceeds, you cannot browse the file.

Add the following code example to the corresponding **HTML** page to render the **Uploadbox** control with the customized file size.

### HTML

```
<div class="control">
<div id="Uploadbox"></div>
</div>
```

Initialize the **Uploadbox** using the following code example.

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
$(function () {
var sample = new ej.Uploadbox($("#Uploadbox"), {
saveUrl: "save.ashx",
removeUrl: "removeFiles.ashx",
error: "error",
fileSize: 1048576 //bytes
});
}
function error(e, ui) {
alert(e.error);
}
```

To know about file action, we need to refer link:

<https://help.syncfusion.com/js/uploadbox/file-actions>

The following screenshot displays **Uploadbox** control with customized file size.

When you want to browse the file within the **fileSize**, you can browse and upload the files.





When you try to browse the file with exceeded fileSize, we cannot browse and upload the files.



### Maximum File Upload Size in IIS

By default, the IIS web server allows for limited file size to be uploaded to the web server. By default, **Machine.config** is configured to accept HTTP Requests up to 4096 KB, that is, 4 MB.

#### *How to upload files up to 28.6 MB?*

In order to allow larger file size uploads, such as above 4MB, you can override it by modifying the “maxRequestLength” attribute in the web application’s configuration file **web.config**. The property *maxRequestLength* indicates the maximum file upload size of 28.6MB, supported by ASP.NET. You cannot upload the files when the **fileSize** property is below the **maxRequestLength** value.

#### Property

| Use-Case Description             | Type     | Maximum request size | Details                                    |
|----------------------------------|----------|----------------------|--|
| <a href="#">maxRequestLength</a> | Property | 28.6 MB              | Maximum request size supported by ASP.NET. |

Add the following code to your **web.config** file.

#### **C#**

```
[web.config]
<configuration>
<system.web>
<httpRuntime maxRequestLength="102400"/> //kilobytes
</system.web>
</configuration>
```

**Note:** maxRequestLength is measured in kilobytes.

#### *How to upload the files above 28.6 MB?*

The **maxRequestLength** property can be increased by modifying the “maxAllowedContentLength” attribute in the web application's configuration file **web.config**. The default maximum length of the content in a request supported by IIS is around 28.6 MB or 30000000bytes.

#### Property

| Use-Case Description                    | Type     | Default value | Details  |
|---|----------|---------------|--|
| <a href="#">maxAllowedContentLength</a> | Property | 28.6 MB       | maxAllowedContentLength specifies the maximum length of content in a request supported by IIS. |

You can add the following code to your **web.config** file in order to set that value to 100 MB.

#### **C#**

```
[web.config]
```

```
<system.webServer>
<security>
<requestFiltering>
<requestLimits maxAllowedContentLength="104857600" /> //bytes
</requestFiltering>
</security>
</system.webServer>
```

---

**Note:** maxAllowedContentLength is measured in bytes.

---



---

**Note:** \* When you configure both maxAllowedContentLength, maxRequestLength attributes, then maxAllowedContentLength is considered for execution.

\* When the upload file's size exceeds maxAllowedContentLength, you get a 404.13 error page.

\* When the upload file's size exceeds maxRequestLength value, you get an exception "System.Web.HttpException: Maximum request length exceeded".

\* The ASP.NET method of maxRequestLength is greater than or equal to the IIS method of limiting the request length (maxAllowedContentLength).

---

### Enables or Disables the Uploadbox

This feature helps set the **enable** or **disable** option for **Uploadbox** by setting **Boolean** type value to **enabled** property. For **enable** or **disable** option, set **enabled** property to '**false**'. The data type is **Boolean**.

The following steps explain the configuration of **enabled** property in **Uploadbox**.

In the **HTML** page, add the **<div>** element to configure the **Uploadbox** element.

#### HTML

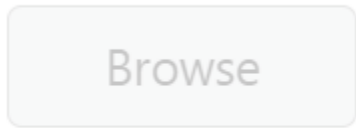
```
<div id="Uploadbox"></div>
```

#### JS

```
// Initialize the control in JavaScript.
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
// Initialize the control in JavaScript.
$(function () {
//Declaration.
var sample = new ej.Uploadbox($("#Uploadbox"), {
saveUrl: "saveFiles.ashx",
removeUrl: "removeFiles.ashx",
enabled: false
});
});
}
```

For **JS**, configure **saveFiles.ashx** and **removeFiles.ashx** files as mentioned in the Save file action and Remove file action respectively.

The following screenshot displays the output.



### Asynchronous Upload

This feature allows you to upload and remove files **asynchronously**. When multiple files are chosen in Asynchronous upload, files will be uploaded one by one to the server. User interaction with the page will not be interrupted at the time of upload. User can also remove the file even after uploading. This is the best way of file upload when compared to synchronous so by default UploadBox works with asynchronous upload option only. To achieve this, set the **asyncUpload** property to **'true'**. The default value of **asyncUpload** property is **'true'**. The data type is **Boolean**.

The following steps guide you in uploading the file **asynchronously**.

In the **HTML** page, add the **<div>** element to configure the **Uploadbox** element.

#### HTML

```
<div id="Uploadbox"></div>
```

#### JS

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
    // Initialize the control in JavaScript.
    $(function () {
        //Declaration.
        var sample = new ej.Uploadbox($("#Uploadbox"), {
            saveUrl: "saveFiles.ashx",
            removeUrl: "removeFiles.ashx",
            asyncUpload: true
        });
    });
}
```

For **JS**, configure **saveFiles.ashx** and **removeFiles.ashx** files as mentioned in the Save file action and Remove file action respectively.

### Synchronous Upload

This features allow you to upload and remove the files **synchronously**. When multiple files are chosen in Synchronous upload, all files will be uploaded only on form submission. Multitasking is not possible here. To achieve this, set the **asyncUpload** property to **'false'**. The data type is **Boolean**.

---

**Note:** By default, Uploadbox widget works with asynchronous upload option only.

---

The following steps guide you in uploading the file **synchronously**.

In the **HTML** page, create a form with action and post method and then add the **<div>** element into the form to configure the **Uploadbox** element.

### HTML

```
<div class="control">
<form id="upload" method="post" action="saveFiles.ashx">
<div id="Uploadbox"></div>
<input type="submit" value="submit" />
</form>
</div>
```

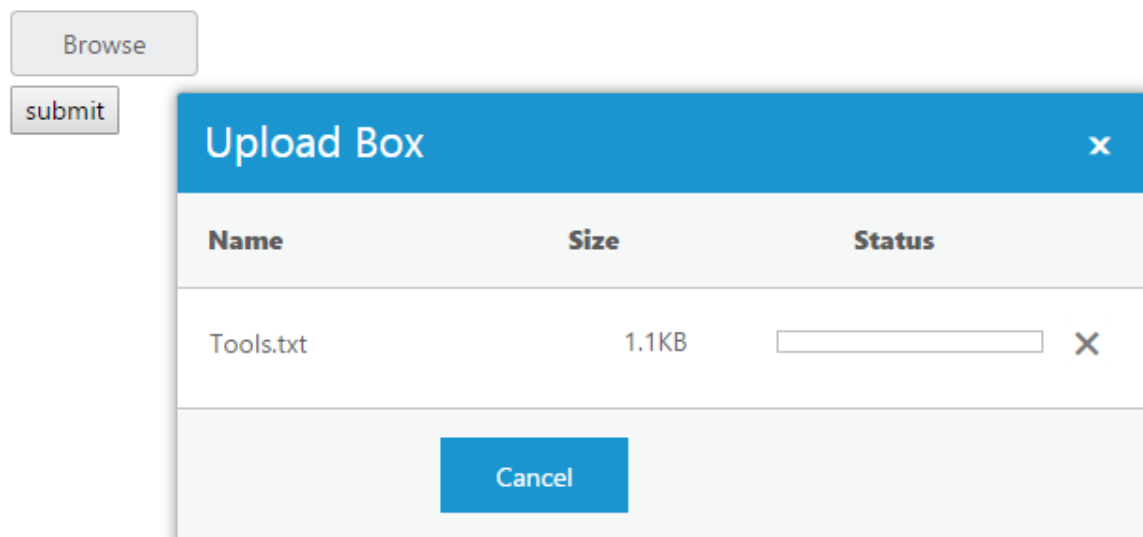
### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
$(function () {
var sample = new ej.Uploadbox($("#Uploadbox"), {
asyncUpload: false
});
});
}
```

For **JS**, configure **saveFiles.ashx** and **removeFiles.ashx** files as mentioned in the Save file action and Remove file action respectively.

Once the form is submitted by using **submit** button, it triggers the **saveFiles.ashx** handler. In the handler, save the files as usual.

The following screenshot displays the output.



## Multiple files upload

The **Uploadbox** widget provides support to upload multiple files spontaneously. The **multipleFilesSelection** property enables you to select multiple files while browsing. To achieve this, set the **multipleFilesSelection** property to 'true'. The data type is **Boolean**.

**Note:** The Multiple file selection supports all the latest versions of browser except Internet Explorer 9 and its previous versions.

The following steps explain the configuration of **multipleFilesSelection** property in **Uploadbox**.

In the **HTML** page, add the **<div>** element to configure the **Uploadbox** element.

### HTML

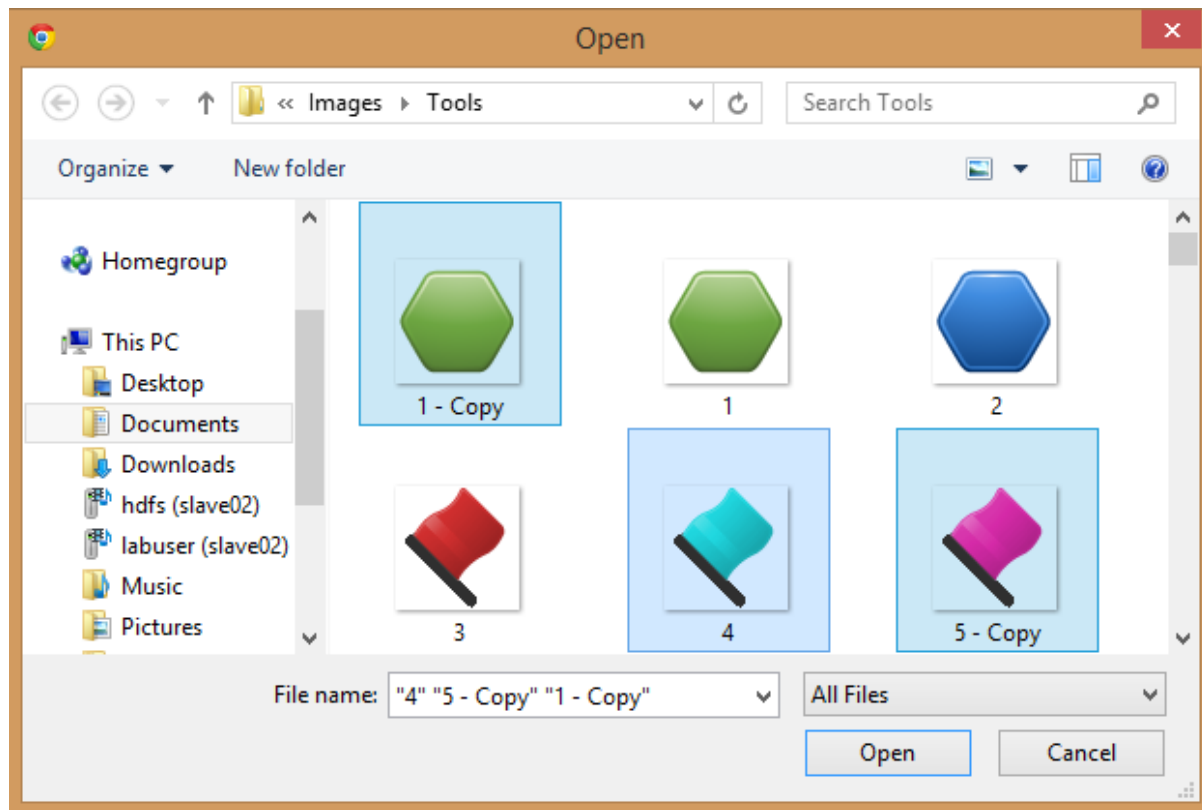
```
<div class="control">
<div id="Uploadbox"></div>
</div>
```

### JS

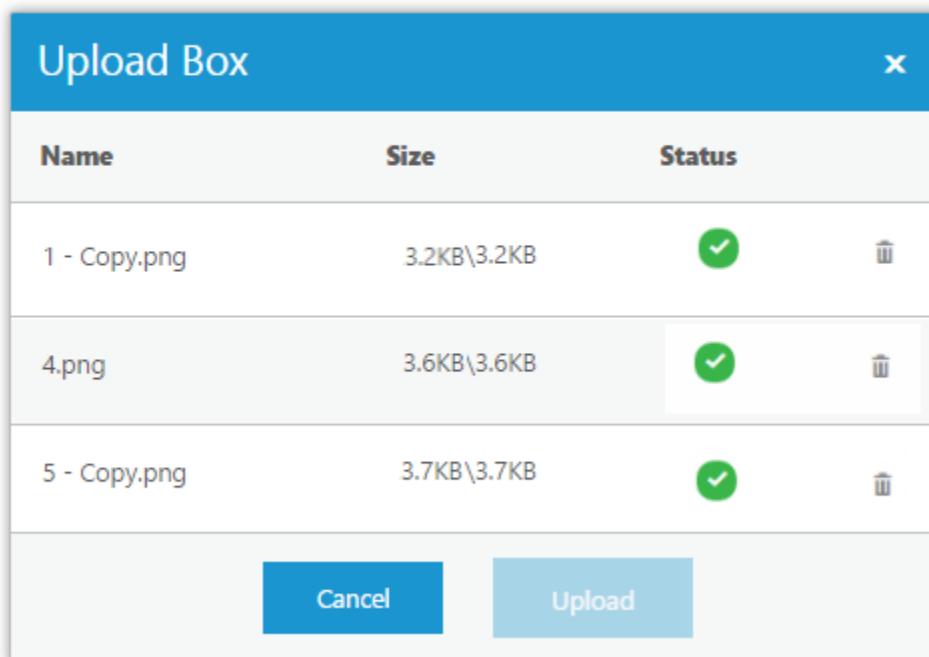
```
// Initialize the control in JavaScript.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
$(function () {
var sample = new ej.Uploadbox($("#Uploadbox"), {
saveUrl: "saveFiles.ashx",
removeUrl: "removeFiles.ashx",
multipleFilesSelection: true
});
});
}
```

For **JS**, configure **saveFiles.ashx** and **removeFiles.ashx** files as mentioned in the Save file action and Remove file action respectively.

The following screenshot displays the output.



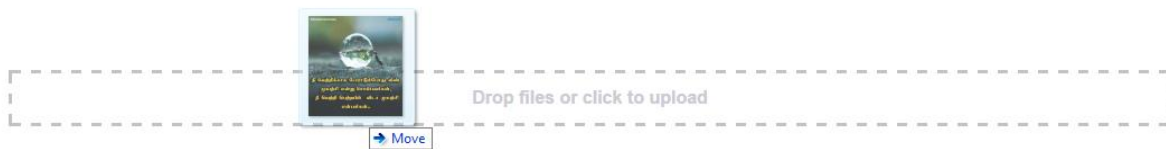
Browse



## Drag and Drop Support

The **Uploadbox** control provides the drag and drop support. You can simply drag-and-drop files, directly from the computer and can be dropped into the droppable area. A list of files can be dragged and dropped when you enable the **multipleFilesSelection**.

The following screenshot displays the drag and drop support.



## Enable drag and drop

You can enable or disable drag and drop by using the **allowDragAndDrop** property. By default, the **allowDragAndDrop** property is set as **False** in the **Uploadbox** control. You can enable drag and drop by setting the **allowDragAndDrop** property as **True**. When you want to drag and drop multiple files, you can enable multiple file selection by setting **multipleFilesSelection** as **True** in the **Uploadbox** control.

The following steps explain how to enable the drag and drop in the **Uploadbox** control.

In the HTML page, add a **<div>** element to enable the drag and drop in Uploadbox control.

### HTML

```
<div class="frame">
<div class="control">
<div id="Uploadbox"></div>
</div>
</div>
```

### JS

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
  // Initialize the control in JavaScript.
  $(function () {
    //Declaration.
    var sample = new ej.Uploadbox($("#Uploadbox"), {
      saveUrl: "http://js.syncfusion.com/demos/web/uploadbox/saveFiles.ashx",
      removeUrl:
        "http://js.syncfusion.com/demos/web/uploadbox/removeFiles.ashx",
      allowDragAndDrop: true,
      multipleFilesSelection: true
    });
  });
}
```

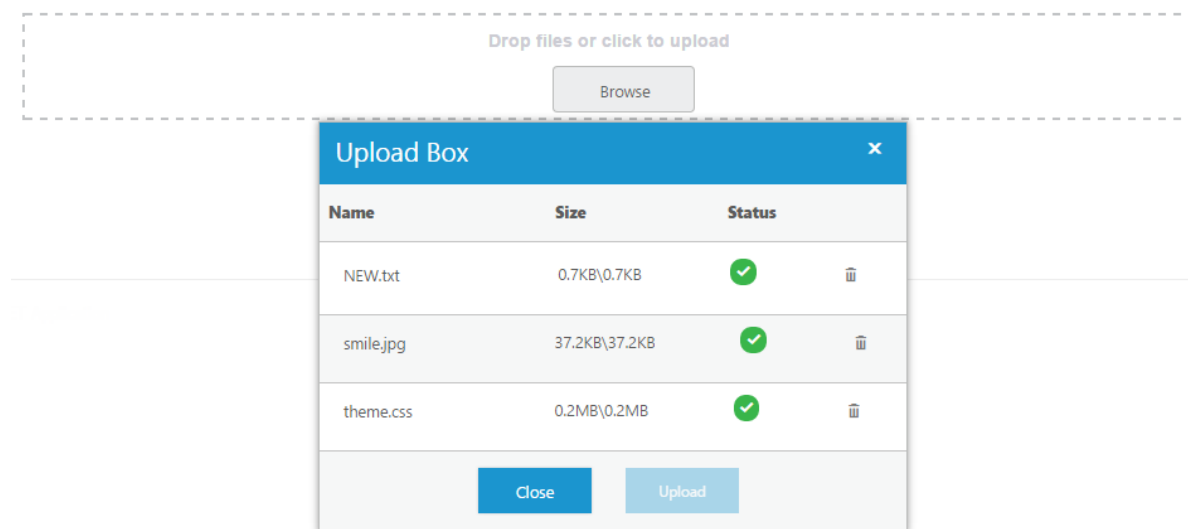
To know about file action, refer to the following link: <https://help.syncfusion.com/js/uploadbox/file-actions>

In CSS, configure the custom styles for drag and drop.

## CSS

```
<style>
.frame {
width: 500px;
height: 100px;
margin-top: 10%;
}
.control {
width: 100%;
height: 100%;
}
</style>
```

The following screenshot displays the output for the above code.



## Drag Area text

You can change the drag area text by using the **dragAreaText** property. By default, the **dragAreaText** (string) property is **Drop files or click to upload** in the Uploadbox control.

In the **HTML** page, add a **<div>** element to enable the drag and drop in the **Uploadbox** control.

## HTML

```
<div class="frame">
<div class="control">
<div id="Uploadbox"></div>
</div>
</div>
```

## JS

```
// Initialize the UploadBox using the following code example.
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
// Initialize the control in JavaScript.
```



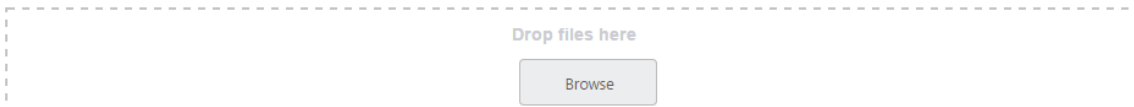
```
$(function () {  
  //Declaration.  
  var sample = new ej.Uploadbox($("#Uploadbox"), {  
    saveUrl: "http://js.syncfusion.com/demos/web/uploadbox/saveFiles.ashx",  
    removeUrl:  
    "http://js.syncfusion.com/demos/web/uploadbox/removeFiles.ashx",  
    allowDragAndDrop: true,  
    dragAreaText: "Drop files here",  
    multipleFilesSelection: true  
  });  
});  
}
```

In CSS, configure the custom styles for drag and drop.

### CSS

```
<style>  
.frame {  
width: 500px;  
height: 100px;  
margin-top: 10%;  
}  
.control {  
width: 100%;  
height: 100%;  
}  
</style>
```

The following screenshot displays the output for the above code.



### Adjust Drop area size

The **Uploadbox** control provides the ability to change or adjust the drop area size. The **dropAreaHeight** and **dropAreaWidth** properties in the **Uploadbox** control allows you to set the maximum height and maximum width for the drop area. The value set to this property is **string** or **number** type.

The following steps explain you on how to adjust the Drop Area Size.

In the **HTML** page, add a **<div>** element to enable the drag and drop in **Uploadbox** control.

### HTML

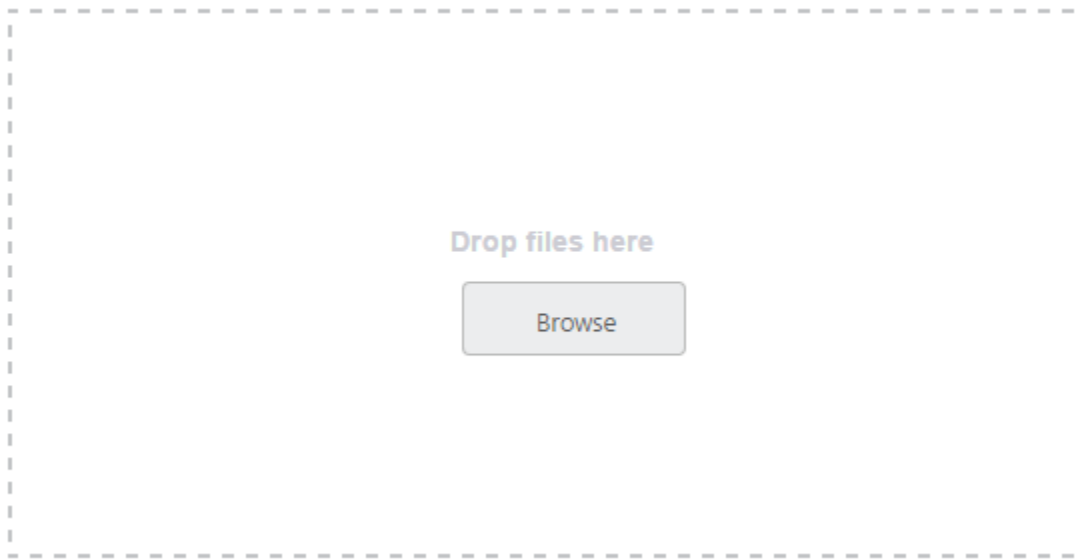
```
<div class="control">  
<div id="Uploadbox"></div>  
</div>
```

### JS

```
/// <reference path="../../tsfiles/jquery.d.ts" />  
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
```

```
module UploadboxComponent {  
  // Initialize the control in JavaScript.  
  $(function () {  
    //Declaration.  
    var sample = new ej.Uploadbox($("#Uploadbox"), {  
      saveUrl: "http://js.syncfusion.com/demos/web/uploadbox/saveFiles.ashx",  
      removeUrl:  
        "http://js.syncfusion.com/demos/web/uploadbox/removeFiles.ashx",  
      allowDragAndDrop: true,  
      multipleFilesSelection: true,  
      dropAreaHeight: "300px",  
      dropAreaWidth: "600px"  
    });  
  });  
}
```

The following screenshot displays the output for the above code.



#### Drop area with Browse button behavior

You can click anywhere in the droppable area to browse and upload the files. The droppable area behaves like a browse button.

#### *Droppable area behavior*

Enable the **allowDragAndDrop** property to achieve this feature. Next, set the **showBrowseButton** as **False** in Uploadbox Control.

The following steps explain the droppable area containing the browse button behavior.

In the **HTML** page, add a **<div>** element to enable drag and drop in the **Uploadbox** control.

#### **HTML**

```
<div class="frame">  
  <div class="control">  
    <div id="Uploadbox"></div>  
  </div>  
</div>
```

## JS

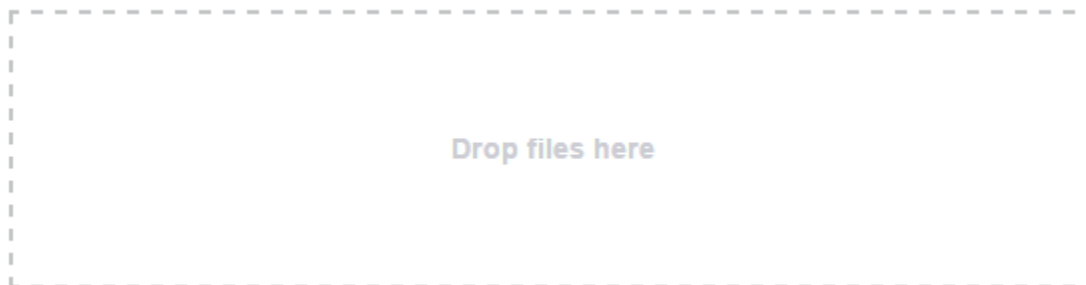
```
// Initialize the Uploadbox by using the following code example.
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
    // Initialize the control in JavaScript.
    $(function () {
        //Declaration.
        var sample = new ej.Uploadbox($("#Uploadbox"), {
            saveUrl: "http://js.syncfusion.com/demos/web/uploadbox/saveFiles.ashx",
            removeUrl:
                "http://js.syncfusion.com/demos/web/uploadbox/removeFiles.ashx",
            allowDragAndDrop: true,
            showBrowseButton: false,
            multipleFilesSelection: true
        });
    });
}
```

In CSS, configure the custom styles for drag and drop.

## CSS

```
<style>
.frame {
width: 500px;
height: 100px;
margin-top: 10%;
}
.control {
width: 100%;
height: 100%;
}
</style>
```

The following screenshot displays the output for the above code.



## Appearance and styling

The **Uploadbox** widget provides support to customize the **dialog box** text and **button** text.

### Customizing Button Text

The following table contains the **sub properties** available under **buttonText** property. To customize the text, pass the alternate text with corresponding **sub properties**.

Sub-properties under buttonText property

| Name   | Description                                  | Data Type |
|--------|--|-----------|
| browse | Sets the alternative text for browse button. | String    |
| upload | Sets the alternative text for upload button. | String    |
| cancel | Sets the alternative text for cancel button. | String    |

The following steps explain the configuration of **buttonText** property in **Uploadbox**.

In the **HTML** page, add the **<div>** element to configure the **Uploadbox** element.

#### HTML

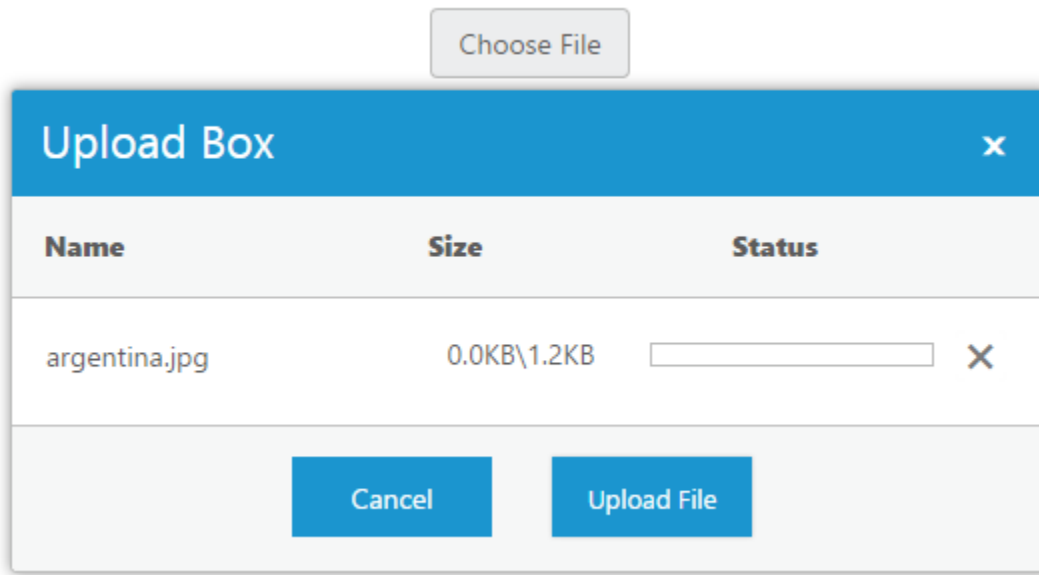
```
<div class="control">
<div id="Uploadbox"></div>
</div>
```

#### JS

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
    // Initialize the control in JavaScript.
    $(function () {
        //Declaration.
        var sample = new ej.Uploadbox($("#Uploadbox"), {
            saveUrl: "saveFiles.ashx",
            removeUrl: "removeFiles.ashx",
            buttonText: { browse: "Choose File", upload: "Upload File", cancel:
                "Cancel Upload" }
        });
    });
}
```

For **JS**, configure **saveFiles.ashx** and **removeFiles.ashx** files as mentioned in the Save file action and Remove file action respectively.

The following screenshot displays the output.



### Customizing Upload Dialog

The following table contains the **sub properties** available under **Uploadbox's DialogText** property. To customize the text, pass the alternate text with corresponding **sub properties**.

Sub properties under dialogText property.

| Name   | Description  |
|--------|--|
| title  | Sets the alternative text for Title of Uploadbox dialog. |
| name   | Sets the alternative text for Name column.               |
| size   | Sets the alternative text for Size column.               |
| status | Sets the alternative text for status column.             |

The following steps explain the configuration of **dialogText** property in **Uploadbox**.

In the **HTML** page, add the **<div>** element to configure the **Uploadbox** element.

#### HTML

```
<div class="control">
  <div id="Uploadbox"></div>
</div>
```

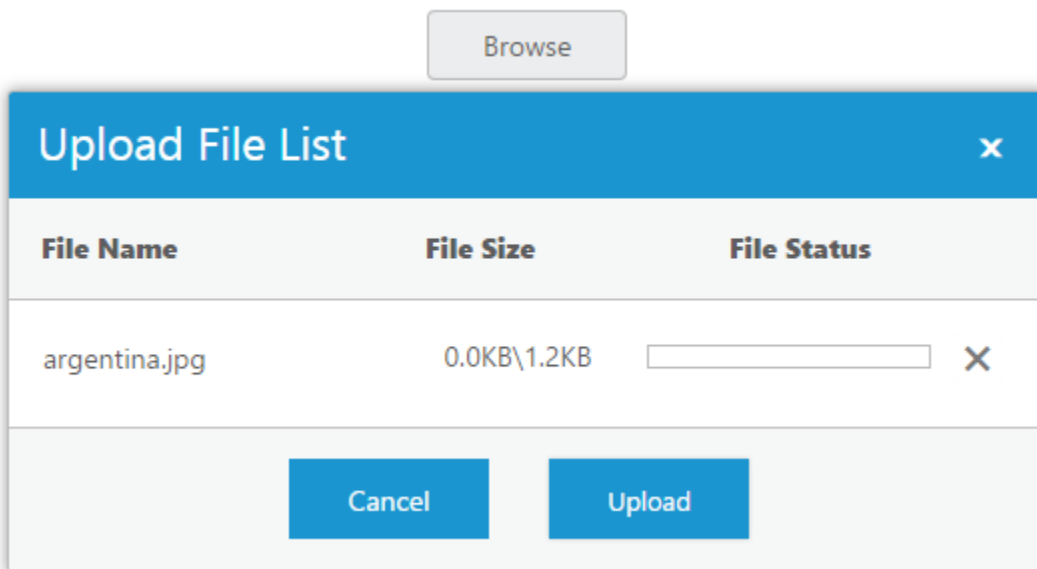
#### JS

```
/// <reference path="../tsfiles/jquery.d.ts" />
/// <reference path="../tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
  // Initialize the control in JavaScript.
  $(function () {
    //Declaration.
```

```
var sample = new ej.Uploadbox($("#Uploadbox"), {
  saveUrl: "saveFiles.ashx",
  removeUrl: "removeFiles.ashx",
  dialogText: { title: "Upload File List", name: "File Name", size: "File Size", status: "File Status" }
});
});
}
```

For **JS**, configure **saveFiles.ashx** and **removeFiles.ashx** files as mentioned in the Save file action and Remove file action respectively.

The following screenshot displays the output.



#### Show or Hide File details

You have an option to **show** or **hide file details** in the **uploaded file list dialog**. By using this property, the **uploaded file dialog** does not display the file details once selected. To enable this, set **showFileDetails** to **'false'**. By default, its value is set to **'true'**. The data type is **Boolean**.

The following steps explain the configuration of **showFileDetails** property in **Uploadbox**.

In the **HTML** page, add the **<div>** element to configure the **Uploadbox** element.

#### HTML

```
<div class="control">
  <div id="Uploadbox"></div>
</div>
```

#### JS

```
/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
  // Initialize the control in JavaScript.
  $(function () {
```

```
//Declaration.  
var sample = new ej.Uploadbox($("#Uploadbox"), {  
  saveUrl: "saveFiles.ashx",  
  removeUrl: "removeFiles.ashx",  
  showFileDetails: false  
});  
});  
}
```

For **JS**, configure **saveFiles.ashx** and **removeFiles.ashx** files as mentioned in the Save file action and Remove file action respectively.

### Theme

**Uploadbox** control's style and appearance are controlled based on **CSS** classes. In order to apply styles to the **Uploadbox** control, you can refer to two files namely, **ej.widgets.core.min.css** and **ej.theme.min.css**. When the file **ej.widgets.all.min.css** is referred, then it is not necessary to include the files **ej.widgets.core.min.css** and **ej.theme.min.css** in your project, as **ej.widgets.all.min.css** is the combination of these both files.

By default, there are 16-theme support available for **Uploadbox** control namely,

- Default-theme
- Flat-azure-dark
- Fat-lime
- Flat-lime-dark
- Flat-saffron
- Flat-saffron-dark
- Gradient-azure
- Gradient-azure-dark
- Gradient-lime
- Gradient-lime-dark
- Gradient-saffron
- Gradient-saffron-dark
- high-contrast-01
- high-contrast-02
- material
- office-365

### Custom CSS

**CSS class** customizes the **Uploadbox** control's appearance. Define a **CSS class** as per the requirement and assign the class name to **cssClass** property. The data type is **string**.

The following steps explain the configuration of **cssClass** property in **Uploadbox**.

In the **HTML** page, add the **<div>** element to configure the **Uploadbox** element.

### HTML

```
<div class="control">  
<div id="Uploadbox"></div>  
</div>
```

**JS**

```

/// <reference path="../../tsfiles/jquery.d.ts" />
/// <reference path="../../tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
    // Initialize the control in JavaScript.
    $(function () {
        //Declaration.
        var sample = new ej.Uploadbox($("#Uploadbox"), {
            saveUrl: "saveFiles.ashx",
            removeUrl: "removeFiles.ashx",
            cssClass: "custom"
        });
    });
}

```

In **CSS**, configure Custom Styles for the **Uploadbox**.

**CSS**

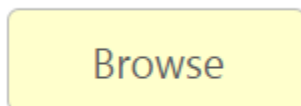
```

<style class="cssStyles">
.custom.e-uploadbox.e-widget .e-selectpart.e-select{
background-color: #FFFFCC;
font-weight: bold;
font-family: sans-serif;
}
</style>

```

For **JS**, configure **saveFiles.ashx** and **removeFiles.ashx** files as mentioned in the Save file action and Remove file action respectively.

The following screenshot displays the output.

**RTL Support**

This feature supports the change of left-to-right alignment of the **Uploadbox** widget to right-to-left (**RTL**). That is, it sets the **Uploadbox** to right-to-left actions.

The following steps explain the configuration of **enableRTL** property in **Uploadbox**.

In the **HTML** page, add the **<div>** element to configure the **Uploadbox** element.

**HTML**

```

<div class="control">
<div id="Uploadbox"></div>
</div>

```

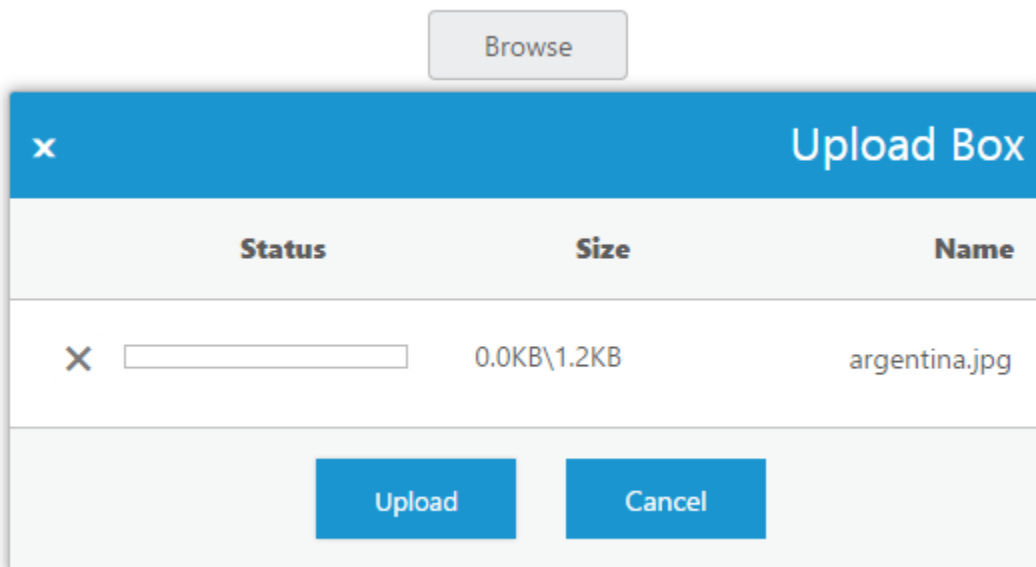
**JS**



```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module UploadboxComponent {
  $(function () {
    var sample = new ej.Uploadbox($("#Uploadbox"), {
      saveUrl: "saveFiles.ashx",
      removeUrl: "removeFiles.ashx",
      enableRTL: true
    });
  });
}
```

For **JS**, configure **saveFiles.ashx** and **removeFiles.ashx** files as mentioned in the Save file action and Remove file action respectively.

The following screenshot displays the output.



## WaitingPopup

### Overview

The **WaitingPopup control for TypeScript** is a visual element that provides support for displaying a pop-up indicator over a target area and preventing the end user's interaction with the target area while loading.

### Key Features

- **Custom text:** Supports custom text inside the pop-up panel.
- **Template:** Supports including HTML content instead of the default image.
- **Transparency:** Supports customizing the transparency and opacity level.
- **Themes:** JavaScript controls include 17 built-in themes (6 flat, 6 gradient, bootstrap, 2 high-contrast, material and office-365 effects) and also support the custom skin option to set user-defined themes.

## Getting Started

This section explains briefly about how to create a **WaitingPopup** in your application with **TypeScript**.

**Essential TypeScript WaitingPopup** provides support to display a **WaitingPopup** within your web page. From the following guidelines, you can learn how to create a **WaitingPopup** in a real-time login page authentication scenario.

The following screenshot illustrates the functionality of a **WaitingPopup** with login page scenario.



You can give the Username and Password in the **login page**. When you click the **Login** button, you get the **WaitingPopup**. After loading, the alert box pops up with the message "Signed in successfully".

### Create Username and Password

**Essential TypeScript WaitingPopup** widget basically renders built-in features like blocking the other actions until the page is loaded. You can easily create the **WaitingPopup** widget by using simple **<div>** element as follows.

Create an HTML file and add the following template to the HTML file.

#### HTML

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Getting Started - RichTextEditor</title>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"></script>
</head>
</html>
```

Add **<div>** element to render a **WaitingPopup**.

#### HTML

```
<div id="targetElement">
<table class="loginTable">
<tr>
<td>Username</td>
<td><input type="text" /></td>
</tr>
<tr>
```

```

<td>Password</td>
<td><input type="password"/></td>
</tr>
<tr>
<td></td>
<td><button id="target">Login</button></td>
</tr>
</table>
<div id="popup"></div>
</div>

```

Initialize **Click function** using the following code example.

### JS

```

/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module WaitingpopupComponent {
$(function () {
$("#target").click(function () {
/*Add waiting popup*/
});
});
}

```

Apply the following styles to show the **WaitingPopup**.

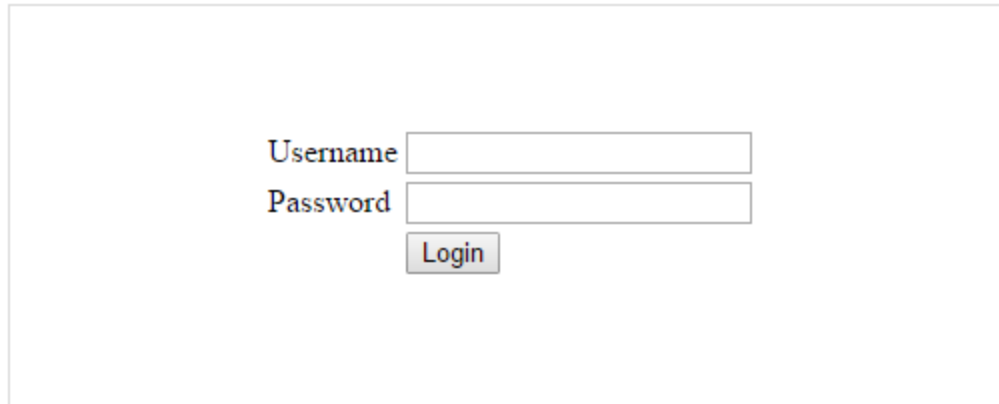
### CSS

```

<style type="text/css" class="cssStyles">
#targetElement {
width: 500px;
height: 200px;
margin: 50px;
border: 1px solid #dbdcdb;
}
.loginTable {
margin: 60px auto;
}
#popup_WaitingPopup .e-image {
display: block;
height: 70px;
}
</style>

```

The following screenshot displays a **User login**.



Username

Password

Login

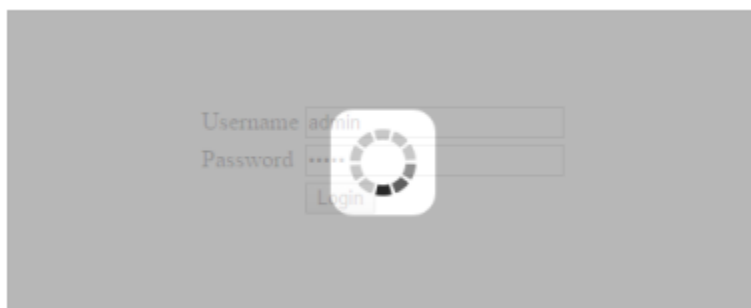
### Add WaitingPopup Widget

In a real-time login page scenario, when you click the Login button, the WaitingPopup is displayed.

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module WaitingpopupComponent {
  $(function () {
    $("#target").click(function () {
      var waiting = new ej.WaitingPopup($("#popup"), {
        showOnInit: true,
        target: "#targetElement"
      });
      function success() {
        alert("Signed in successfully");
        waiting.hide();
      }
      setTimeout(success, 5000);
    });
  });
}
```

The following screenshot shows the output of the above code example.



### Behavior and Settings

#### Automatic Initializing WaitingPopup widget

**WaitingPopup** widget contains **showOnInit** property that allows the popup to display over a target on page load automatically. By default, **showOnInit** property is set as false.

The following steps explain you on how to display the **WaitingPopup** on page load.

In an **HTML** page, add a **<div>** element to render **WaitingPopup** widget.

### HTML

```
<div class="control">
<div id="waitingPopUp"></div>
</div>
```

### JS

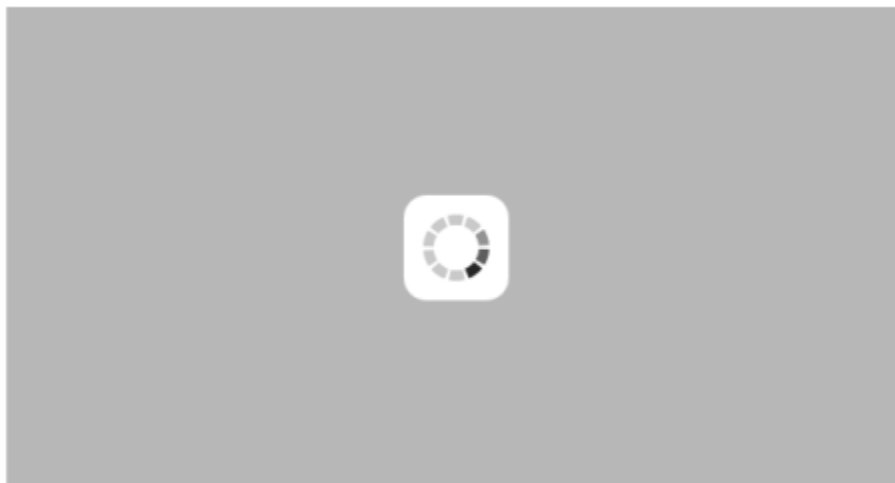
```
// Configure the WaitingPopup to display automatically on page load as follows.
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module WaitingPopupComponent {
$(function () {
var sample = new ej.WaitingPopup($("#waitingPopUp"), {
showOnInit: true,
});
});
}
```

Add the following styles to render **WaitingPopup** widget.

### CSS

```
<style type="text/css" class="cssStyles">
#waitingPopUp {
height: 320px;
width: 600px;
}
</style>
```

The following screenshot illustrates the **WaitingPopup** when **showOnInit** is set to **"true"**.



### Enable / Disable Popup Indicator

You can show or hide the popup indicator of **WaitingPopup** widget using **showImage** property. By default, **showImage** property is set as **true**.

The following steps explains you to enable / disable popup indicator in **WaitingPopup** widget.

In the **HTML** page, add a **<div>** element to render **WaitingPopup** widget.

#### HTML

```
<div class="control">
<div id="waitingPopUp"></div>
</div>
```

#### JS

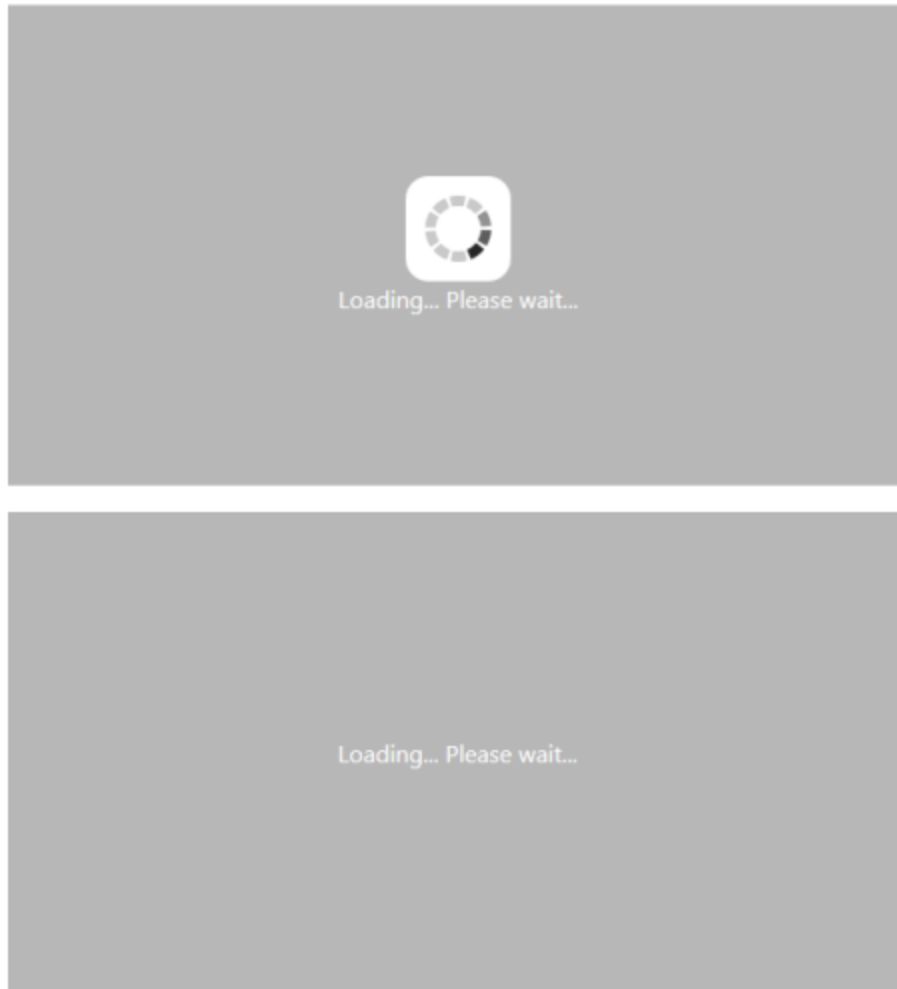
```
// To configure Enable / Disable popup indicator in WaitingPopup, use the
following code.
//Enable popup indicator:
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module WaitingPopupComponent {
$(function () {
var sample = new ej.WaitingPopup($("#waitingPopUp"), {
showOnInit: true,
showImage: true,
text: "Loading... Please wait..."
});
});
}
//Disable popup indicator:
$(function () {
// declaration
var sample = new ej.WaitingPopup($("#waitingPopUp"), {
showOnInit: true,
showImage: false,
text: "Loading... Please wait..."
});
});
});
```

Add the following styles to render **WaitingPopup** widget.

#### CSS

```
<style type="text/css" class="cssStyles">
#waitingPopUp {
height: 320px;
width: 600px;
}
</style>
```

Execute the above code to render the following output.



### Show / Hide WaitingPopup

Using **show()** and **hide()** methods, you can display or hide the **WaitingPopup** widget over the target area.

The following steps explain you to show / hide the **WaitingPopup** widget.

In the **HTML** page, add a **<div>** element to render **WaitingPopup** widget.

#### HTML

```
<div class="control">
<div id="waitingPopUp"></div>
</div>
```

#### JS

```
// Use the following code to Show / Hide WaitingPopup.
//Show WaitingPopup:
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module WaitingPopupComponent {
$(function () {
var sample = new ej.WaitingPopup($("#target"), {
```

```

var popUpObj = $("#waitingPopUp").data("ejWaitingPopup");
popUpObj.show();
});
}
//Hide WaitingPopup:
$(function () {
var sample = new ej.WaitingPopup($("#target"), {
var popUpObj = $("#waitingPopUp").data("ejWaitingPopup");
popUpObj.hide();
});
});

```

Add the following styles to render **WaitingPopup** widget.

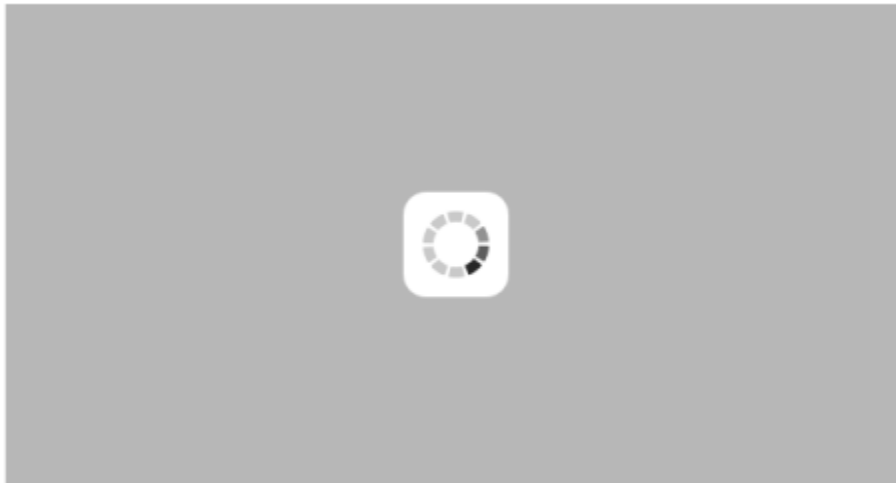
### CSS

```

<style type="text/css" class="cssStyles">
#waitingPopUp {
height: 320px;
width: 600px;
}
</style>

```

The following screenshot illustrates a **WaitingPopup** when **show()** method is invoked.



## Appearance and Styling

### Custom Text

**WaitingPopup** control provides support for Custom Text to mention any message inside the pop-up panel. You can specify a custom text through the option **text** that displays when the **WaitingPopup** is loading.

The following steps explain you the configuration of the custom text for **WaitingPopup** control.

In the **HTML** page, add a **<div>** element to render **WaitingPopup** widget.

### HTML

```

<div class="control">
<div id="waitingPopUp"></div>
</div>

```



## JS

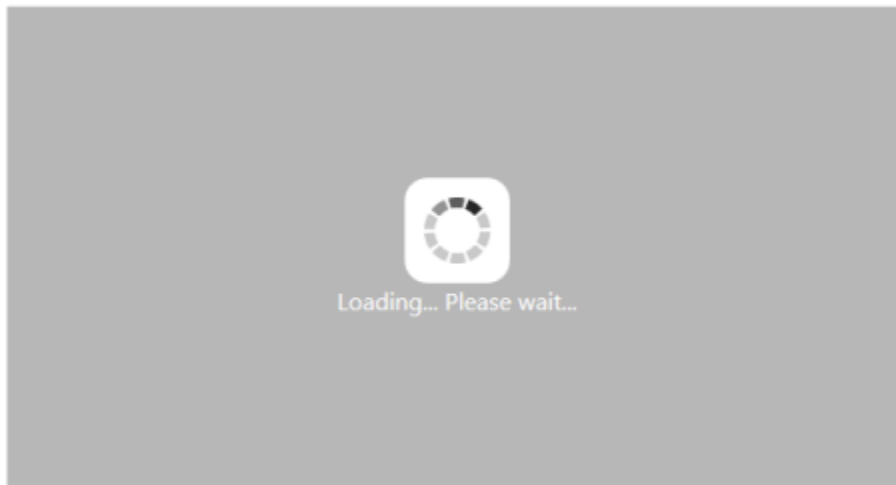
```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module WaitingPopupComponent {
  $(function () {
    // Add Custom Text for WaitingPopup control as follows.
    // declaration
    var sample = new ej.WaitingPopup($("#waitingPopUp"), {
      showOnInit: true,
      text: "Loading... Please wait..."
    });
  });
}
```

Add the following styles to render **WaitingPopup** widget.

## CSS

```
<style type="text/css" class="cssStyles">
#waitingPopUp {
height: 320px;
width: 600px;
}
</style>
```

Execute the above code to render the following output.



## Template

**WaitingPopup** widget provides support for template to customize the appearance of it by including **HTML** content instead of the default image.

The following steps explain you on how to define template to display a text and image for **WaitingPopup** control.

In the **HTML** page, add a **<div>** element to configure **WaitingPopup** widget.

## HTML

```
<div class="control">
<div id="waitingPopUp"></div>
</div>
```

Add customized template for **WaitingPopup** control using the following code example.

### HTML

```
<div id="content">
<div class="block">
<div class="logo"></div>
<div class="txt">
<p>Create cutting edge </p>
<p><b>HTML5</b> web application </p>
<p>with ease </p>
</div>
</div>
<div class="loader"></div>
</div>
```

Initialize the **WaitingPopup** with custom template using the following code example.

### JS

```
/// <reference path="tsfiles/jquery.d.ts" />
/// <reference path="tsfiles/ej.web.all.d.ts" />
module WaitingPopupComponent {
$(function () {
var sample = new ej.WaitingPopup($("#waitingPopUp"), {
// declaration
showOnInit: true,
template: $("#content")
});
});
}
```

In **CSS**, you can configure the custom styles for **WaitingPopup**.

**Note:** Images for this sample are available 'installed sample location /images/waitingPopup' and we need to define images in mentioned CSS. Henceforth the images will display.

### CSS

```
<style type="text/css" class="cssStyles">
#waitingPopUp {
height: 320px;
width: 600px;
margin: 0 auto;
}
.block {
height: 76px;
}
.logo {
background-image: url("../images/waitingPopup/js_logo.png");
float: left;
height: 100%;
```

```
width: 77px;
margin-right: 15px;
}
.txt {
float: left;
font-size: 17px;
height: 100%;
text-align: left;
}
.txt p {
margin: 0;
}
.loader {
background: url("../images/waitingPopup/load_light.gif") no-repeat scroll
-5px 18px transparent;
height: 40px;
width: 100%;
}
#content {
cursor: default;
height: 112px;
width: 275px;
}
</style>
```

Execute the above code to render the following output.



### CSS Class

You can use the **CSS** class to customize the **WaitingPopup** control appearance. Define a **CSS** class as per requirement and assign the class name to **cssClass** property.

The following steps allows you to configure **CSS** class for an auto-complete textbox.

In the **HTML** page, add a **<div>** element to configure **WaitingPopup** widget.

### HTML

```
<div class="control">
<div id="waitingPopUp"></div>
</div>
```

## JS

```
// Add the cssClass property to WaitingPopup widget as follows.  
/// <reference path="tsfiles/jquery.d.ts" />  
/// <reference path="tsfiles/ej.web.all.d.ts" />  
module WaitingPopupComponent {  
    $(function () {  
        var sample = new ej.WaitingPopup($("#waitingPopUp"), {  
            showOnInit: true,  
            cssClass: "customStyle",  
            text: "Loading.. Please wait..."  
        });  
    });  
}
```

Define CSS class for customizing the WaitingPopup widget.

## CSS

```
<style type="text/css" class="cssStyles">  
/*Customize the panel property*/  
#waitingPopUp {  
    height: 320px;  
    width: 600px;  
    margin: 0 auto;  
}  
/* Customize the WaitingPopup */  
.customStyle{  
    background-color:darkred;  
    font-style:italic;  
    font-weight:bolder;  
    opacity:0.5;  
}  
</style>
```

The following screenshot displays the output for the above code.

